

HP-UX Reference
Release 11.0
Miscellaneous, Device Files, and Glossary
Sections 5, 7, 9 and Index

Volume 5 of 5

Edition 1



B2355-90166

E1097

Printed in: United States
© Copyright 1997 Hewlett-Packard Company

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright Notices. ©Copyright 1983-1997 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

©Copyright 1979, 1980, 1983, 1985-93 Regents of the University of California

This software is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

©Copyright 1980, 1984, 1986 Novell, Inc.
©Copyright 1986-1992 Sun Microsystems, Inc.
©Copyright 1985, 1986, 1988 Massachusetts Institute of Technology.
©Copyright 1989-1993 The Open Software Foundation, Inc.
©Copyright 1986 Digital Equipment Corporation.
©Copyright 1990 Motorola, Inc.
©Copyright 1990-1995 Cornell University
©Copyright 1989-1991 The University of Maryland
©Copyright 1988 Carnegie Mellon University
©Copyright 1991-1997 Mentat, Inc.
©Copyright 1996 Morning Star Technologies, Inc.
©Copyright 1996 Progressive Systems, Inc.
©Copyright 1997 Isogon Corporation

Trademark Notices. UNIX is a registered trademark in the United States and other countries, licensed exclusively through The Open Group.

X Window System is a trademark of the Massachusetts Institute of Technology.

MS-DOS and Microsoft are U.S. registered trademarks of Microsoft Corporation.

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the U.S. and other countries.

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. the manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: October 1997 (HP-UX Release 11.0)

Volume Five

Table of Contents

Section 5

Section 7

Section 9

Index

Volume Five
Table of Contents

Section 5
Section 7
Section 9

Index

Table of Contents

Volume Five

Section 5: Miscellaneous Topics

Entry Name(Section): name	Description
intro(5)	introduction to miscellany
acl(5): acl	introduction to access control lists
aio(5): aio()	POSIX asynchronous I/O
aliases(5): aliases	aliases file for sendmail
ascii(5): ascii	map of ASCII character set
audio(5): audio	audio tools available through HP VUE
audit(5)	introduction to HP-UX Auditing System
curses(5): curses	definition for screen handling and optimisation functions
dirent(5): dirent.h	format of directory streams and directory entries
dld.sl(64): dld.sl	dynamic loader
environ(5): environ	user environment
fcntl(5)	file control options
fenv(5): fenv, acosf()	floating-point environment macros and functions
fs_wrapper(5)	configuration and binary files used by file system administration commands
hier(5): hier	file system hierarchy
hostname(5): hostname	host name resolution description
inttypes(5): inttypes	basic integer data types
ioctl(5): ioctl	generic device control commands
lang(5): lang	description of supported languages
langinfo(5): langinfo	language information constants
libcrash(5): libcrash	crash dump access library
limits(5): limits	implementation-specific constants
man(5): man	macros for formatting manpages
manuals(5)	current list of orderable HP-UX documentation
math(5): math	math functions and constants
mknod(5): mknod	macros for handling device numbers
mm(5): mm	the MM macro package for formatting documents
mman(5)	memory mapping definitions
mtab: mounted file system table	see pfs_fstab(5)
ndir(5): ndir.h	format of HP-UX directory streams
nlio(5): nlio	Native Language I/O (NLIO) Subsystem
pam_unix(5): pam_unix	authentication, account, session, and password management PAM modules for UNIX
pam_updb(5): pam_updb	PAM user policy definition module
pd_att(5)	index of attribute manpages for HP Distributed Print Service
pd_att_document(5)	list of attributes for a document object (HP Distributed Print Service)
pd_att_ivdocument(5)	list of attributes for an initial value document object (HP Distributed Print Service)
pd_att_ivjob(5)	list of attributes for an initial value job object (HP Distributed Print Service)
pd_att_job(5)	list of attributes for a job object (HP Distributed Print Services)
pd_att_log(5)	list of attributes for a log object (HP Distributed Print Service)
pd_att_log_ptr(5)	list of attributes for a logical printer object (HP Distributed Print Services)
pd_att_phy_ptr(5)	list of attributes for a physical printer object (HP Distributed Print Service)
pd_att_queue(5)	list of attributes for a queue object (HP Distributed Print Service)
pd_att_spooler(5)	list of attributes for a spooler object (HP Distributed Print Service)
pd_att_supervisor(5)	list of attributes for a supervisor object (HP Distributed Print Service)
pfs_exports(5): pfs_exports	directories to export to PFS clients
pfs_fstab(5): pfs_fstab	static file system mounting table
pfs_xtab(5): pfs_xtab	directories to export to PFS clients see pfs_exports(5)
portal(5): portal.h	header file for future applications
quota(5): quota	disk quotas
rcsintro(5): rcsintro	description of RCS commands
regexp(5): <regexp.h>	regular expression and pattern matching notation definitions
sd(5)	create, distribute, install, monitor, and manage software
signal(5): signal.h	description of signals
signal.h: description of signals	see signal(5)
sis(5)	secure internet services description

Table of Contents

Volume Five

Entry Name(Section):	name	Description
stat(5):	stat	data returned by stat/fstat/lstat system call
stdarg(5):	stdarg	handle variable argument list
stdsyms(5):	stdsyms	description of HP-UX header file organization
suffix(5):	suffix	file-name suffix conventions
term(5):	term	terminal capabilities
types(5)		primitive system data types
unctrl(5):	unctrl	definition for unctrl()
unistd(5):	unistd.h	standard structures and symbolic constants
unistd.h:	standard structures and symbolic constants	see unistd(5)
values(5)		machine-dependent values
varargs(5):	varargs.h	handle variable argument list
x_open(5)		pointer to manual entry for X/Open Conformance Statement Questionnaire

Section 7: Device (Special) Files

Entry Name(Section):	name	Description
intro(7)		introduction to device special files
arp(7P):	arp	address resolution protocol
autochanger(7):	autochanger	SCSI media changer device drivers
blmode(7):	blmode	terminal block mode interface
cent(7):	cent	Centronics-compatible interface
clone(7)		open a major and minor device pair on a STREAMS driver
console(7):	console	system console interface
ct(7):	ct	cartridge tape access
ddfa(7):	ddfa	Data Communications and Terminal Controller Device File Access software
diag0(7):	diag0	diagnostic interface to I/O subsystem
disk(7):	disk	direct disk access
fddi(7)		Fiber Distributed Data Interface Tools
floppy(7):	floppy	flexible or "floppy" disk device driver
framebuf(7):	framebuf	information for raster frame-buffer devices
hil(7):	hil	HP-HIL device driver
hilkbd(7):	hilkbd	HP-HIL mapped keyboard driver
hpib(7):	hpib	Hewlett-Packard Interface Bus driver
inet(7F):	inet	Internet protocol family
iomap(7):	iomap	physical address mapping
IP(7P):	IP	Internet Protocol
kmem:	kernel memory	see mem(7)
kmem(7):	kmem	perform I/O on kernel memory based on symbol name
lan(7):	lan	network I/O card access information
ldterm(7):	ldterm	STREAMS terminal line discipline module
lp(7):	lp	line printer
lvm(7):	lvm	Logical Volume Manager (LVM)
mem(7):	mem, kmem	main memory
modem(7):	modem	asynchronous serial modem line control
mt(7):	mt	magnetic tape interface for stape, tape1, and tape2
nfs(7):	nfs, NFS	network file system
null(7):	null	null file
pckt(7):	pckt	Packet Mode module for STREAMS pty
ps2(7):	ps2, ps2kbd, ps2mouse	PS/2 keyboard and mouse device driver and files
ptem(7):	ptem	STREAMS pty (pseudo-terminal) Emulation module
ptm(7):	ptm	STREAMS master pty (pseudo-terminal) driver
pts(7):	pts	STREAMS slave pty driver
pty(7):	pty	pseudo terminal driver
routing(7)		system support for local network packet routing
sad(7)		STREAMS administrative driver
scsi(7):	scsi	Small Computer System Interface (SCSI) device drivers
scsi_ctl(7):	scsi_ctl	SCSI pass-through device driver
scsi_disk(7):	scsi_disk	SCSI direct access device driver
scsi_pt(7):	scsi_pt	SCSI pass-through device driver
scsi_tape(7):	scsi_tape	SCSI sequential access device driver
socket(7):	socket	Interprocess communications

Entry Name(Section): name	Description
streamio(7)	STREAMS ioctl commands
strlog(7)	STREAMS log driver
sttyv6(7): stty	terminal interface for Version 6/PWB compatibility
TCP(7P): TCP	Internet Transmission Control Protocol
telm: STREAMS Telnet master driver	see tels(7)
tels(7): tels, telm	STREAMS slave and master drivers
termio(7): termio, termios	general terminal interface
termios: general terminal interface	see termio(7)
termiox(7): termiox	extended general terminal interface
timod(7)	STREAMS module for reads and writes by Transport Interface users
tirdwr(7)	STREAMS module for reads and writes by Transport Interface users
tty(7): tty	controlling terminal interface
UDP(7P): udp	Internet user datagram protocol
UNIX(7P): UNIX	local communication domain protocol
vxfsio(7)	VxFS file system control functions
xopen_networking(7): xopen_networking	Interprocess communications

Section 9: General Information

Entry Name(Section): name	Description
intro(9)	introduction to the general information section
glossary(9)	a description of common HP-UX terms
Introduction(9)	an introduction to the HP-UX operating system and the HP-UX Reference

Index

Notes

Section 5

Miscellaneous Topics

Section 5

Miscellaneous Topics

NAME

intro - introduction to miscellany

DESCRIPTION

This section describes miscellaneous facilities such as macro packages, character set tables, and the file system hierarchy.

SEE ALSO

Introduction(9).

NAME

acl - introduction to access control lists

DESCRIPTION

Access control lists are a key enforcement mechanism of discretionary access control (see Definitions below), for specifying access to files by users and groups more selectively than traditional HP-UX mechanisms allow.

HP-UX already enables non-privileged users or processes, such as file owners, to allow or deny other users access to files and other objects on a “need to know” basis, as determined by their user and/or group identity (see *passwd(4)* and *group(4)*). This level of control is accomplished by setting or manipulating a file's permission bits to grant or restrict access by owner, group, and others (see *chmod(2)*).

ACLs offer a greater degree of selectivity than permission bits. ACLs allow the file owner or superuser to permit or deny access to a list of users, groups, or combinations thereof.

ACLs are supported as a superset of the UNIX operating system discretionary access control (DAC) mechanism for files, but not for other objects such as inter-process communication (IPC) objects.

Definitions

Because control of access to data is a key concern of computer security, we provide the following definitions, based on those of the *Department of Defense Trusted Computer System Evaluation Criteria*, to explain further both the concepts of access control and its relevance to HP-UX security features:

access “A specific type of interaction between a subject and an object that results in the flow of information from one to the other.” Subjects include “persons, processes, or devices that cause information to flow among objects or change the system state.” Objects include files (ordinary files, directories, special files, FIFOs, etc.) and inter-process communication (IPC) features (shared memory, message queues, semaphores, sockets).

access control list (ACL)

An access control list is a set of (*user.group, mode*) entries associated with a file that specify permissions for all possible user-ID/group-ID combinations.

access control list (ACL) entry

An entry in an ACL that specifies access rights for one user and group ID combination.

change permission

The right to alter DAC information (permission bits or ACL entries). Change permission is granted to object (file) owners and to privileged users.

discretionary access control (DAC)

“A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) to any other subject.”

mode Three bits in each ACL entry which represent read, write, and execute/search permissions. These bits may exist in addition to the 16 mode bits associated with every file in the file system (see *glossary(9)*).

privilege The ability to ignore access restrictions and change restrictions imposed by security policy and implemented in an access control mechanism. In HP-UX, superusers and members of certain groups (see *privgrp(4)*) are the only privileged users.

restrictive versus permissive

An individual ACL entry is considered restrictive or permissive, depending on context. Restrictive entries deny a user and/or group access that would otherwise be granted by less-specific base or optional ACL entries (see below). Permissive entries grant a user and/or group access that would otherwise be denied by less-specific base or optional ACL entries.

Access Control List Entries

An access control list (ACL) consists of sets of (*user.group, mode*) entries associated with a file that specify permissions. Each entry specifies for one user-ID/group-ID combination a set of access permissions, including read, write, and execute/search.

To help understand the relationship between access control lists and traditional file permissions, consider the following file and its permissions:

```
-rwxr-xr-- james admin datafile
```

The file owner is user **james**.

The file's group is **admin**.

The name of the file is **datafile**.

The file owner permissions are **rwX**.

The file group permissions are **r-X**.

The file other permissions are **r--**.

In an ACL, user and group IDs can be represented by names or numbers, found in **/etc/passwd**. The following special symbols can also be used:

- % Symbol representing no specific user or group.
- @ Symbol representing the current file owner or group.

Base ACL Entries

When a file is created, three base access control list entries are mapped from the file's access permission bits to match a file's owner and group and its traditional permission bits. Base ACL entries can be changed by the *chmod(2)* and *setacl(2)* system calls.

- (*uid*%,*mode*) Base ACL entry for the file's owner
- (%,*gid*,*mode*) Base ACL entry for the file's group
- (%,%,*mode*) Base entry for other users

(Except where noted, examples are represented in short form notation. See ACL Notation, below.)

Optional ACL entries

Optional access control list entries contain additional access control information, which the user can set with the *setacl(2)* system call to further allow or deny file access. Up to thirteen additional user/group combinations can be specified.

For example, the following optional access control list entries can be associated with our file:

- (*mary*.*admin*, *rwX*) Grant read, write, and execute access to user **mary** in group **admin**.
- (*george*%, *--*) Deny any access to user **george** in no specific group.

ACL Notation

Supported library calls and commands that manage ACLs recognize three different symbolic representations:

- operator form For input of entire ACLs and modifications to existing ACLs, in a syntax similar to that used by *chmod(1)*.
- short form Easier to read, intended primarily for output. *chacl(1)* accepts this form as input so that it can interpret output from *lsacl(1)*.
- long form A multi-line format useful for greater clarity, and supported only for output.

For our example file, the base ACL entries could be represented in the three notations as follows:

- operator form *james.% = rwX, %.admin = rX, %.% = r*
- short form (*james.%*,*rwX*) (*%.admin*,*r-X*) (*%.%,r--*)
- long form

<i>rwX</i>	<i>james.%</i>
<i>r-X</i>	<i>%.admin</i>
<i>r--</i>	<i>%.%</i>

In addition to basic ACL usage, some library calls and commands understand and use a variation of operator and short forms. See the section below on *ACL Patterns*.

ACL Uniqueness

Entries are unique in each ACL. There can only be one (*u.g, mode*) entry for any pair of *u* and *g* values; one (*u.%*, *mode*) entry for a given value of *u*; one (*%.g*, *mode*) entry for a given value of *g*; and one (*%.%, mode*) entry for each file. For example, an ACL can have a (23.14, *mode*) entry and a (23.%, *mode*) entry, but not two (23.14, *mode*) entries or two (23.%, *mode*) entries.

Access Check Algorithm

ACL entries can be categorized by four levels of specificity. In access checking, ACLs are compared to the effective user and group IDs in this order:

(<i>u.g</i> , <i>rw</i> x)	specific user, specific group
(<i>u.%</i> , <i>rw</i> x)	specific user, no specific group
(<i>%g</i> , <i>rw</i> x)	no specific user, specific group
(<i>%.%</i> , <i>rw</i> x)	no specific user, no specific group

Once an entry for the combination of a process effective user ID and effective group ID (or any supplementary group ID) is matched, no further (that is, less specific) entries are checked. More specific entries that match take precedence over any less specific ones that also match.

If a process has more than one group ID (that is, a non-null supplementary groups list), more than one (*u.g*, *mode*) or (*%g*, *mode*) entry might apply for that process. If so, the access modes in all matching entries (of the same level of specificity, *u.g* or *%g*) are OR'd together. Access is granted if the resulting mode bits allow it. Since entries are unique, the order of entries in each entry type is insignificant.

Because the traditional UNIX permission bits are mapped into base ACL entries, they are included in access checks.

If a request is made for more than one type of access, such as opening a file for both reading and writing, access is granted only if the process is allowed all requested types of access. Note that access can be granted if the process has two groups in its groups list, one of which is only allowed read access, and the other of which is only allowed write access. In other words, even if the requested access is not granted by any one entry, it may be granted by a combination of entries due to the process belonging to several groups.

Operator Form of ACLs (input only)

user. *group* *operator* *mode* [*operator* *mode*] ... , ...

Multiple entries are separated by commas, as in *chmod*(1). Each entry consists of a user identifier and group identifier followed by one or more operators and mode characters, as in the mode syntax accepted by *chmod*(1).

The entire ACL must be a single argument, and thus should be quoted to the shell if it contains whitespace or special characters. Whitespace is ignored except within names. A null ACL is legitimate, and means either "no access" or "no changes", depending on context.

Each user or group ID may be represented by:

<i>name</i>	Valid user or group name.
<i>number</i>	Valid numeric ID value.
%	"No specific user or group," as appropriate.
@	"Current file owner or group," as appropriate; useful for referring to a file's <i>u.%</i> and <i>%g</i> base ACL entries.

An operator is always required in each entry. Operators are:

- = Set all bits in the entry to the given mode value.
- + Set the indicated mode bits in the entry.
- Clear the indicated mode bits in the entry.

The mode is represented by an octal value of 0 through 7; or any combination of **r**, **w**, and **x** can be given in any order (see EXAMPLES below). A null mode denies access if the operator is =, or represents "no change" if the operator is + or -.

Multiple entries and multiple operator-mode parts in an entry are applied in the order specified. Conflicts do not result in error; the last specified entry or operator takes effect. Entries need not appear in any particular order.

Note that *chmod*(1) allows only **u**, **g**, **o**, or **a** to refer symbolically to the file owner, group, other, or all users, respectively. Since ACLs work with arbitrary user and group identifiers, @ is provided as a convenience.

The exact syntax is:

```

acl ::= [entry[,entry]...]
entry ::= id . id op mode [op mode]...
id ::= name | number | % | @
op ::= = | + | -

```

```

mode ::= 0..7 | [char[char]...]
char ::= r | w | x

```

Short Form of ACLs (input and output)

(user . group, mode) ...

Short form differs from operator form in several ways:

- Entries are surrounded by parentheses rather than being separated by commas.
- Each entry specifies the mode, including all mode bits. It is not possible to change the mode value with + and - operators. However, the comma functions like the = operator in operator form.
- For clarity, hyphens represent unset permission bits in the output of the mode field and are allowed in input. This resembles the mode output style used by *ls*(1).

Multiple entries are concatenated. For consistency with operator form, a dot (.) is used to separate user and group IDs.

On output, no whitespace is printed except in names (if any). ID numbers are printed if no matching names are known. Either ID can be printed as % for “no specific user or group.” The mode is represented as <r|-><w|-><x|->, that is, it always has three characters, padded with hyphens for unset mode bits. If the ACL is read from the system, entries are ordered by specificity, then by numeric values of ID parts.

On input, the entire ACL must be a single argument, and thus should be quoted to the shell if it contains whitespace or special characters. Whitespace is ignored except within names. A null ACL is legitimate, and means either “no access” or “no changes”, depending on context.

User and group IDs are represented as in operator form.

The mode is represented by an octal value of 0 through 7; or any combination of r, w, x and - (ignored) can be given in any order (see EXAMPLES below). A null mode denies access.

Redundancy does not result in error; the last entry for any user-ID/group-ID combination takes effect. Entries need not appear in any particular order.

The exact syntax is:

```

acl ::= [entry[entry]...]
entry ::= (id.id,mode)
id ::= name | number | % | @
mode ::= 0..7 | [char[char]...]
char ::= r | w | x | -

```

Long Form of ACLs (output only)

mode user . group

Each entry occupies a single line of output. The mode appears first in a fixed-width field, using hyphens (for unset mode bits) for easy vertical scanning. Each user and group ID is shown as a name if known, a number if unknown, or % for “no specific user or group.” Entries are ordered from most to least specific, then by numeric values of ID parts.

Note that every ACL printed has at least three entries, the base ACL entries (that is, *uid*%, *%gid*, and *%%*).

The exact syntax is:

```

acl ::= entry[<newline>entry]...
entry ::= mode<space>id.id
mode ::= <r|-><w|-><x|->
id ::= name | number | %

```

ACL Patterns

Some library calls and commands recognize and use ACL patterns instead of exact ACLs to allow operations on all entries that match the patterns. ACL syntax is extended in the following ways:

wildcard user and group IDs

A user or group name of * (wildcard) matches the user or group ID in any entry, including % (no specific user or group).

mode bits on, off, or ignored

For operator-form input, the operators =, +, and - are applied as follows:

- = entry mode value matches this mode value exactly
- + these bits turned on in entry mode value
- these bits turned off in entry mode value

When only + and - operators are used, commands ignore the values of unspecified mode bits.

Short-form patterns treat the mode identically to the = operator in operator form.

wildcard mode values

A mode of * (wildcard) in operator or short form input (for example, "ajs.%=*" or "(ajs.%,*)") matches any mode value, provided no other mode value is given in a operator-form entry. Also, the mode part of an entry can be omitted altogether for the same effect.

entries not combined

Entries with matching user and group ID values are not combined. Each entry specified is applied separately by commands that accept patterns.

ACL Operations Supported

The system calls *setacl(2)* and *getacl(2)* allow setting or getting the entire ACL for a file in the form of an array of *acl_entry* structures. To check access rights to a file, see *access(2)* and *getaccess(2)*.

Various library calls are provided to manage ACLs:

acltostr(3C) Convert *acl_entry* arrays to printable strings.

strtoacl(3C) Parse and convert ACL strings to *acl_entry* arrays.

strtoaclpatt(3C) Parse and convert ACL pattern strings to *acl_entry_patt* arrays.

setaclentry(3C)

fsetaclentry Add, modify, or delete a single ACL entry in one file's ACL.

cpacl(3C)

fcpac Copy an ACL and file miscellaneous mode bits (see *chmod(2)*) from one file to another, transfer ownership if needed (see below), and handle remote files correctly.

chownacl(3C) Change the file owner and/or group represented in an ACL, that is, transfer ownership (see below).

The following commands are available to manage ACLs and permissions:

chacl(1) Add, modify, or delete individual entries or all optional entries in ACLs on one or more files, remove all access to files, or incorporate ACLs into permission bits.

lsacl(1) List ACLs on files.

chmod(1) Change permission bits and other file miscellaneous mode bits.

ls(1) In long form, list permission bits and other file attributes.

find(1) Find files according to their attributes, including ACLs.

getaccess(1) List access rights to file(s).

ACL Interaction with *stat(2)*, *chmod(2)*, and *chown(2)*

stat The *st_mode* field summarizes the caller's access rights to the file. It differs from file permission bits only if the file has one or more optional entries applicable to the caller. The *st_basemode* field provides the file's actual permission bits. The *st_acl* field indicates the presence of optional ACL entries in the file's ACL.

The *st_mode* field contains a user-dependent summary, so that programs ignorant of ACLs that use *stat(2)* and *chmod(2)* are more likely to produce expected results, and so that *stat(2)* provides reasonable information about remote files over NFS. The *st_basemode* and *st_acl* fields are useful only for local files.

chmod For conformance with IEEE Standard POSIX 1003.1-1988, *chmod(2)* deletes any optional entries in a file's ACL. Unfortunately, since *chmod(2)* is used to set file miscellaneous mode bits as well as permission bits, extra effort is required in some cases to preserve a file's ACL.

chown If the new owner and/or group of a file does not already have an optional (*u.%*, *mode*) and/or (*%g*, *mode*) entry in the file's ACL, it inherits the old owner's and/or group's file access permission bits and base ACL entry:

```
(id1,mode1) -> (id2,mode1)
```

This is the traditional behavior. However, if the new owner and/or group of a file already has an optional (*u.%*, *mode*) and/or (*%g*, *mode*) entry in the file's ACL, the ACL does not change:

```
(id1, mode1) -> (id1, mode1)
(id2, mode2) -> (id2, mode2)
```

Existing access information in the ACL is preserved. However, because the old optional ACL entry becomes the new base ACL entry and vice versa, the file's access permission bits change.

Transferring ownership of ACLs by *chown(2)* allows a file to be transferred to a different user or group, or copied by a different user or group than the owner (using *cpacl(3C)* or *chownacl(3C)*), and later returned to the original owner or group without net changes to its ACL. The extra complexity is necessary because:

- ACLs are a backward-compatible superset of permission bits (which are coupled to file owner and group IDs), not a replacement for them.
- it enables users and programs that deal with ACLs to do so simply, rather than with a combination of permission bits and ACL entries. Also, the access check algorithm is simpler and more symmetrical; permission bits do not “eclipse” or “mask” ACL entries.

EXAMPLES

Operator Form

The following sets the *%%* entry to restrict “other” users to only reading the file.

```
chacl '%.% = r' myfile
```

The following allows user “bill” in any group to write the file, assuming that no restrictive entry is more specific than the **bill.%** entry (for example, a **bill.adm** entry that denies writing).

```
chacl 'bill.% +w' myfile
```

The following ACL specification contains two entries. The first one deletes write and adds read capability to the entry for user 12, group 4. The second entry denies access for any unspecified user in any unspecified group.

```
chacl '12.4-w+r, %.% =' myfile
```

The following pair of entries sets the *u.%* entry for the file's owner to allow both read and execute and results in adding write and execute capabilities for “other” users (the “*%%*” entry). Note that a mode character is purposely repeated for illustration purposes.

```
chacl '@.% = 5, %.% + xwx' myfile
```

Short Form

Here is a typical ACL as it might be printed. It allows user **jpc** to read or execute the file while in group **adm**; it denies user **ajs** access to the file while in group **trux**; it allows user **jpc** in any group (except **adm**) to only read the file; any other user in group **bin** may read or execute the file; and any other user may only read the file.

```
(jpc.adm,r-x)(ajs.trux,---)(jpc.%,r--)(%.bin,r-x)(%.%,r--)
```

The following allows “other” users to only read the file.

```
chacl '(%.,r)' myfile
```

The following sets write-only access for user **bill** in any group.

```
chacl '(bill.%, -w-)' myfile
```

The following sets the entry for user 12 in group 4 to allow read and write.

```
chacl '(12.4,wr)' myfile
```

The following sets the base ACL entry for the file's owner to allow both read and execute, and sets write and execute capabilities for “other” users (the “*%%*” entry).

```
chacl '@.%, 5) (%.%, xwx)' myfile
```

Long Form

Here is the same ACL as in an earlier example, printed in long form.

```
r-x   jpc.adm
---   ajs.trux
r--   jpc.%
r-x   %.bin
r--   %.%
```

ACL Patterns

The following command locates files whose ACLs contain an entry that allows read access and denies write access to some user/group combination.

```
find / -acl '*.*+r-w' -print
```

The following matches entries for any user in group **bin** and for user **tammy** in any group, regardless of the entries' mode values. Matching optional ACL entries are deleted and mode values in matching base ACL entries are set to zero:

```
chacl -d '%.bin, tammy.*=*' myfile
```

The following matches all entries, deleting optional entries and setting mode values of base ACL entries to zero:

```
chacl -d '(*.*,*)' myfile
```

HEADERS

Header <sys/acl.h>

The <sys/acl.h> header file defines the following constants to govern the numbers of entries per ACL:

NACLENTRIES	maximum number of entries per ACL, including base entries
NBASEENTRIES	number of base entries
NOPTENTRIES	number of optional entries

The ACL entry structure **struct acl_entry** is also defined, and includes the following members:

```
aclid_t   uid;      /* user ID */
aclid_t   gid;      /* group ID */
aclmode_t mode;     /* see <unistd.h> */
```

The <sys/acl.h> header also defines the types **aclid_t** and **aclmode_t**.

Non-specific user and group ID values:

ACL_NSUSER	non-specific user ID
ACL_NSGROUP	non-specific group ID

A special *nentries* value ACL_DELOPT is used with *setacl(2)* to delete optional entries.

Header <sys/getaccess.h>

The <sys/getaccess.h> header defines constants for use with *getaccess(2)*.

Special parameter values for *uid*:

UID_EUID	use effective user ID
UID_RUID	use real user ID
UID_SUID	use saved user ID

Special parameter values for *ngroups*:

NGROUPS_EGID	process's effective gid
NGROUPS_RGID	process's real gid
NGROUPS_SGID	process's saved gid
NGROUPS_SUPP	process's supplementary groups only
NGROUPS_EGID_SUPP	process's eff gid plus supp groups
NGROUPS_RGID_SUPP	process's real gid plus supp groups
NGROUPS_SGID_SUPP	process's saved gid plus supp groups

Header <acllib.h>

The <acllib.h> header file defines several constants for use with ACL support library calls.

Symbolic forms of ACLs for *acltostr()*:

```
FORM_SHORT
FORM_LONG
```

Magic values for various calls:

```
ACL_FILEOWNER    file's owner ID
ACL_FILEGROUP    file's group ID
ACL_ANYUSER      wildcard user ID
ACL_ANYGROUP     wildcard group ID
MODE_DEL         delete one ACL entry
```

Mask for valid mode bits in ACL entries:

```
MODEMASK    (R_OK | W_OK | X_OK)
```

The <acllib.h> header also defines the **struct acl_entry_patt** ACL pattern entry structure, which includes the following members:

```
aclid_t      uid;          /* user ID */
aclid_t      gid;          /* group ID */
aclmode_t    onmode;       /* mode bits that must be on */
aclmode_t    offmode;      /* mode bits that must be off */
```

WARNINGS

ACLs are intended for use on ordinary files and directories. Optional ACL entries are not recommended on files that are manipulated by certain system utilities, such as terminal special files and LP scheduler control files. These utilities might delete optional entries, including those whose intent is restrictive, without warning as a consequence of calling *chmod(2)*, thereby increasing access unexpectedly.

Most, but not all, supported utilities are able to handle ACLs correctly. However, only the *fbbackup(1M)* and *frecover(1M)* file archive utilities handle access control lists properly. When using programs (such as archive programs *ar(1)*, *cpio(1)*, *ftio(1)*, *tar(1)*, and *dump(1M)*) unable to handle ACLs on files with optional ACL entries, note the Access Control List information included on their respective reference pages, to avoid loss of data.

If a user name is defined in the */etc/passwd* file or a group name is defined in the */etc/group* file as % or @, or for patterns, *, ACL syntax cannot reference that name as itself because the symbols have other meanings. However, such users or groups can still be referenced by their ID numbers. User and/or group names must not include the following characters:

- . Do not use in user names.
- + Do not use in group names.
- Do not use in group names.
- = Do not use for operator form input of group names.
- , Do not use for short form or for operator form patterns.
-) Do not use for short form patterns.

It is possible to specify an ACL pattern using the @ (file owner or group) or * (wildcard) symbols so that it cannot match certain files, perhaps depending on their ownership, by giving two entries, one with specific values and the other using @ or *, which are equivalent for a file but contain different mode values. For example:

```
find / -acl '(ajs.%,r)(@.%,rw)' -print
```

cannot match a file owned by **ajs**.

DEPENDENCIES

NFS NFS does not support ACLs on remote files. Individual manual entries specify the behavior of various system calls, library calls, and commands under these circumstances. Be careful when transferring a file with optional entries over a network or when manipulating a remote file because optional entries may be silently deleted.

AUTHOR

The access control list design described here was developed by HP.

FILES

<code><sys/acl.h></code>	Header file that supports <i>setacl(2)</i> and <i>getacl(2)</i> .
<code><sys/getaccess.h></code>	Header file that supports <i>getaccess(2)</i> .
<code><acllib.h></code>	Header file that supports ACL library calls.
<code>/etc/passwd</code>	Defines user names and user and group ID values.
<code>/etc/group</code>	Defines group names.

SEE ALSO

chacl(1), chmod(1), cp(1), find(1), getaccess(1), ln(1), ls(1), lsacl(1), mv(1), rm(1), fbackup(1M), frecover(1M), fsck(1M), fsdb(1M), access(2), chmod(2), chown(2), creat(2), getaccess(2), getacl(2), mknod(2), open(2), setacl(2), stat(2), acltostr(3C), chownacl(3C), cpacl(3C), setaclentry(3C), strtocl(3C), group(4), passwd(4), privgrp(4).

NAME

aio - POSIX asynchronous I/O facility

SYNOPSIS

```
#include <aio.h>
```

DESCRIPTION

The POSIX Asynchronous I/O facility implements Section 6.7 of IEEE Standard 1003.1b-1993, Standard for Information Technology, Portable Operating System Interface (POSIX), Part 1: System Application Program Interface (API), Amendment 1: Realtime Extensions (C Language). It allows a process (thread) to start multiple simultaneous read and/or write operations to multiple files, wait for or obtain notification of completion of requested operations, and to retrieve the status of completed operations. The purpose of the POSIX Asynchronous I/O facility is to allow a process (thread) to overlap some elements of computation and information processing with I/O processing.

Interface Functions

The POSIX Asynchronous I/O facility includes the following interface functions:

```
aio_read()    Start an asynchronous read operation
aio_write()   Start an asynchronous write operation
lio_listio()  Start a list of asynchronous I/O operations
aio_suspend() Wait for completion of one or more asynchronous I/O operations
aio_error()   Retrieve the error status of an asynchronous I/O operation
aio_return()  Retrieve the return status of an asynchronous I/O operation and free any associated system resources
aio_cancel()  Request cancellation of a pending asynchronous I/O operation
aio_fsync()   Request synchronization of the media image of a file to which asynchronous operations have been addressed
```

To use these functions, link in the realtime library by specifying `-lrt` on the compiler or linker command line.

Asynchronous I/O Control Block

The Asynchronous I/O Control Block (**aiocb**) is used as a parameter to all of the asynchronous I/O functions. The **aiocb** specifies parameters for an asynchronous I/O operation in a call to **aio_read()**, **aio_write()**, or **lio_listio()** and then may be used as a "handle" for the enqueued asynchronous I/O operation in a subsequent call to **aio_cancel()**, **aio_suspend()**, **aio_fsync()**, **aio_error()**, or **aio_return()**.

The **aiocb** structure is defined in `#include <aio.h>` as:

```
struct aiocb {
    int      aio_fildes;          /* IEEE Std 1003.1b, Section 6.7.1.1 */
    off_t    aio_offset;          /* file descriptor */
    volatile void *aio_buf;       /* file offset */
    size_t    aio_nbytes;         /* location of buffer */
    int      aio_reqprio;         /* length of transfer */
    struct sigevent aio_sigevent; /* request priority offset */
    int      aio_lio_opcode;       /* signal number and value */
    /* operation to be performed */
};
```

The **aiocb** supplied to **aio_read()**, **aio_write()**, or **lio_listio()** must contain the parameters that would be supplied in a normal synchronous **read()** or **write()** function call, where **aio_fildes** corresponds to **fildes**, **aio_nbytes** corresponds to **nbytes**, **aio_offset** corresponds to the implicit file offset. The **aiocb** may also contain request priority delta and signaling information to satisfy unique realtime and asynchronous I/O requirements. For the **lio_listio()** function, the **aio_lio_opcode** field specifies whether the operation is a read or write.

Once an asynchronous I/O operation has been enqueued for a particular **aiocb** its address is used as a handle for other asynchronous I/O functions and can only be used to refer to a single enqueued operation.

The fields `aio_offset64_pad`, `aio_return`, `aio_errno`, and `reserved` are also defined in the `aio_cb` structure but are reserved for future use and extension. They are all ignored by the asynchronous I/O facility.

Manifest Constants

Certain values as defined by the POSIX standard are declared in `aio.h`.

The following values are returned by the `aio_cancel()` function:

- AIO_CANCELED** All specified asynchronous I/O operations were successfully canceled.
- AIO_NOTCANCELED** At least one specified asynchronous I/O operations was not successfully canceled.
- AIO_ALLDONE** All specified asynchronous I/O operations were completed before the request was processed.

The following values are valid values of the `flags` field that controls return from the `lio_listio()` function:

- LIO_WAIT** Wait for all specified operations to complete.
- LIO_NOWAIT** Return without waiting for operations to complete.

The following values are operation codes supplied in the `aio_lio_opcode` field the designate the type of an operation started with `lio_listio()`.

- LIO_READ** The `aio_cb` specifies an asynchronous read operation.
- LIO_WRITE** The `aio_cb` specifies an asynchronous write operation.
- LIO_NOP** The `aio_cb` specifies no operation and is silently ignored.

Enqueuing of Operations

If an error condition is detected that prevents an operation from being started, `aio_read()` and `aio_write()` do not enqueue a request. Instead they immediately return `-1` and set `errno` to indicate the cause of the failure. Once an operation has been successfully enqueued, calls to `aio_error()` and `aio_return()` referencing the `aio_cb` referred to by `aio_cbp` must be used to determine the status of the operation and any error conditions, including those normally reported by `read()` and `write()`. The request remains enqueued and consumes process and system resources until `aio_return()` is called.

Error reporting of operations enqueued by `lio_listio()` may be less immediate than that of operations enqueued by `aio_read()` and `aio_write()`. With the exception of resource shortages, errors for which `aio_read()` and `aio_write()` would immediately return `-1` and an `errno` value do not cause `lio_listio()` to stop enqueueing the current or subsequent requests in its list. Instead, partial success occurs. In this case, the application must use `aio_error()` to determine which operations in its list have been enqueued and which resulted in errors.

Asynchronous I/O operations are said to be *complete* when:

- The I/O transfer is performed successfully.
- An error is detected in one or more parameters of the operation.
- The operation is canceled. If a valid `sigevent` is specified in the `aio_cb` is used to start the operation, then the signal is delivered when the operation completes.

Reading and Writing Asynchronously

Asynchronous read and write operations are started using the `aio_read()`, `aio_write()`, and `lio_listio()` interfaces. The parameters for each operation are provided in the `aio_cb` used to start the operation. A list of `aio_cb` pointers can be provided to the `lio_listio()` function call, in which case the type (read or write) of the operation is determined from the `aio_lio_opcode` field of the `aio_cb`. Once started, the I/O operations may proceed concurrently with execution of the process (thread) that initiated the operation.

With the implementation of process threads in the HP-UX implementation, an application may achieve asynchronous I/O behavior by using synchronous `read()` and `write()` functions from independent threads within the process. However, the application may have to implement many of the status management facilities provided in the POSIX Asynchronous I/O facility.

Waiting for Completion

The POSIX Asynchronous I/O facility supports both *polling* and *notification* models. The *polling* model is implemented by repeatedly calling the `aio_error()` function to test the status of an operation. The *notification* model is implemented by designating a *sigevent* in the `aioctx` used to start the operation. The specified notification, if any, is then performed when the operation completes.

The `aio_suspend()` function allows the application to wait for completion of one or more asynchronous I/O operations. A *timeout* may be set so that the process can again execute and take appropriate recovery actions if the expected operations do not complete as expected. If the `aio_suspend()` references multiple operations, return is made when any one of the operations completes.

Retrieving Errors

Once an asynchronous I/O operation has been started, its status can be tested with the `aio_error()` and `aio_return()` functions, which return the current status of a referenced `aioctx`. For a *polling* implementation, the `aio_error()` function is used to check the status until a completion status is seen, then `aio_return()` is used to free the `aioctx` for re-use.

For a *notification* implementation, status of the completed I/O can be determined and the `aioctx` freed with a single call to `aio_return()`.

The errors reported by `aio_error()` and `aio_return()` include all of the errors normally reported by `read()` and `write()` plus errors unique to asynchronous I/O processing. After an asynchronous I/O operation is started but before an error is detected or the operation completes successfully, `aio_error()` will return `EINPROGRESS`.

Cancellation

The `aio_cancel()` function allows an application to request cancellation of an asynchronous I/O operation. The `aioctx` used to start the operation may be used as a handle to identify it for cancellation. Cancellation of all operations pending for a file may also be requested. Not all asynchronous I/O operations can be canceled.

Synchronizing Permanent Storage

The `aio_fsync()` function supports synchronizing the contents of permanent storage when multiple asynchronous I/O operations are outstanding for the file or device. *Only* those requests already enqueued for the designated file at the time of the function call are included in the synchronization operation.

File Offsets

Asynchronous I/O operations are not inherently sequential. Each operation must specify an offset, and the file offset is never updated as a result of an asynchronous I/O operation.

Although correct behavior is supported, it should be noted that setting the `O_APPEND` flag on a file to which asynchronous I/O operations limits the value of asynchronous I/O. When `O_APPEND` is set, operations on the file must be handled serially with the ending file length after one request providing the starting offset for the next. While there may be some advantage to allowing the system to queue requests, care should be taken not to exhaust system or process thread resources by enqueueing a large number of requests that must be processed serially.

System Limitations and Restrictions

The operation of the POSIX Asynchronous I/O interfaces is subject to certain system limitations and restrictions.

Since each enqueued asynchronous I/O operation requires allocation of system memory for its internal control structure, the number of simultaneously enqueued asynchronous I/O operations that the system can have pending is limited. The maximum number of asynchronous I/O operations that all active processes on the system may have enqueued concurrently is tunable. The current maximum value can be obtained using the `sysconf()` call with the argument `_SC_AIO_MAX`. The default maximum value is 2048. In addition to the system-wide limit, there is a per-process limit. It is controlled using the argument `RLIMIT_AIO_OPS` to the `getrlimit()` and `setrlimit()` system calls. Even though an asynchronous I/O operation has completed, it still remains enqueued until `aio_return()` is called for that operation.

The asynchronous I/O operations which use request and call back mechanism for I/O, are subject to a system-wide limit on the amount of physical memory that can be locked during asynchronous I/O transfers. This system-wide maximum number of bytes of memory that can be locked simultaneously for aio requests is tunable. This can be set as a percentage of the physical memory available on the system. By default, it is set to 10% of the physical memory. In addition to the system-wide limit, there is a per-process limit which

is controlled using the argument `RLIMIT_AIO_MEM` to the `getrlimit()` and `setrlimit()` system calls. Further, the amount of memory that can be locked at a time for any reason, not just for asynchronous I/O, is controlled by the system-wide limit `lockable_mem`. Other system activity, including explicit memory locking with `plock()` and/or `mlock()` interfaces may affect the amount of lockable memory at any given time.

a

The maximum priority change that can be specified in `aio_reqprio` is limited. The maximum priority change value is tunable. The current maximum value can be obtained using the `sysconf()` call with the argument `_SC_AIO_PRIO_DELTA_MAX`. The default value is 20.

The maximum number of asynchronous I/O operations that can be specified in a single `lio_listio()` call is limited. This limit is tunable. The current maximum value can be obtained using the `sysconf()` call with the argument `_SC_AIO_LISTIO_MAX`. The default maximum value is 256.

Some asynchronous I/O operations are also subject to both system-wide and per-process limits on the number of simultaneously active threads. See *pthread(3T)*.

Programming Limitations and Restrictions

Altering the contents of or deallocating memory associated with the `aiocb` referred to by `aiocbp` or the buffer referred to by `aiocbp->aio_buf` while an asynchronous read operation is outstanding, that is `aio_return()` has not been called for the `aiocb`, may produce unpredictable results.

EXAMPLES

The following code sequence illustrates an asynchronous read operation and polling for completion.

```
#include <fcntl.h>
#include <errno.h>
#include <aio.h>
char buf[4096];
int retval;
struct aiocb myaiocb;
bzero( &myaiocb, sizeof (struct aiocb));
myaiocb.aio_fildes = open( "/dev/null", O_RDONLY);
myaiocb.aio_offset = 0;
myaiocb.aio_buf = (void *) buf;
myaiocb.aio_nbytes = sizeof (buf);
myaiocb.aio_sigevent.sigev_notify = SIGEV_NONE;
retval = aio_read( &myaiocb );
if (retval) perror("aio:");
/* continue processing */
...
/* wait for completion */
while ( (retval = aio_error( &myaiocb) ) == EINPROGRESS) ;
/* free the aiocb */
retval = aio_return( &myaiocb);
```

SEE ALSO

`aio_cancel(2)`, `aio_error(2)`, `aio_fsync(2)`, `aio_read(2)`, `aio_return(2)`, `aio_suspend(2)`, `aio_write(2)`, `fdatasync(2)`, `fsync(2)`, `getrlimit(2)`, `lio_listio(2)`, `read(2)`, `setrlimit(2)`, `write(2)`, *pthread(3T)*.

STANDARDS CONFORMANCE

aio: POSIX Realtime Extensions, IEEE Std 1003.1b

NAME

aliases - aliases file for sendmail

SYNOPSIS

aliases

DESCRIPTION

This file describes user ID aliases used by **/usr/sbin/sendmail**. The file resides in **/etc/mail** and is formatted as a series of lines of the form

name: name1, name2, name3,...

The *name* is the name to alias, and the *namen* are the aliases for that name. Lines beginning with white space are continuation lines. Lines beginning with # are comments.

Aliasing occurs only on local names. Loops can not occur, since no message will be sent to any person more than once.

After aliasing has been done, local and valid recipients who have a **.forward** file in their home directory have messages forwarded to the list of users defined in that file.

This is only the raw data file; the actual aliasing information is placed into a binary format in the file **/etc/mail/aliases.db** using the program *newaliases(1)*. A **newaliases** command should be executed each time the aliases file is changed for the change to take effect. Note that the NIS alias maps are generated by ypmake using the makemap program, which leaves aliases.pag and aliases.dir in the /etc/mail directory.

SEE ALSO

newaliases(1M), **sendmail(1M)**.

HISTORY

The **aliases** file format appeared in 4.0BSD.

a

NAME

ascii - map of ASCII character set

SYNOPSIS

```
cat /usr/share/lib/pub/ascii
```

DESCRIPTION

/usr/share/lib/pub/ascii provides a map of the ASCII character set, giving both octal and hexadecimal equivalents of each character, to be printed as needed. The file contains the following text:

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq	006 ack	007 bel
010 bs	011 ht	012 nl	013 vt	014 np	015 cr	016 so	017 si
020 dle	021 dc1	022 dc2	023 dc3	024 dc4	025 nak	026 syn	027 etb
030 can	031 em	032 sub	033 esc	034 fs	035 gs	036 rs	037 us
040 sp	041 !	042 "	043 #	044 \$	045 %	046 &	047 ^
050 (051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [134 \	135]	136 ^	137 _
140 `	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174	175 }	176 ~	177 del

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 \$	25 %	26 &	27 ^
28 (29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [5c \	5d]	5e ^	5f _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c	7d }	7e ~	7f del

Control Characters

The following table shows the set of ASCII control characters with their octal, decimal, and hexadecimal values. To obtain the respective characters from the keyboard, use the indicated keypress combinations.

To place control characters in a file when using the **vi** or **ex** editor, type Ctrl-v before typing the desired control character.

Oct	Dec	Hex	Disp	Symbol	Char Name	Keypress
000	000	00	<i>none</i>	NUL	Null	Ctrl-Shift-@
001	001	01	^A	SOH	Start of Header	Ctrl-A
002	002	02	^B	STX	Start of Text	Ctrl-B
003	003	03	^C	ETX	End of Text	Ctrl-C
004	004	04	^D	EOT	End of Transmission	Ctrl-D
005	005	05	^E	ENQ	Enquire	Ctrl-E
006	006	06	^F	ACK	Acknowledge	Ctrl-F
007	007	07	^G	BEL	Bell	Ctrl-G
010	008	08	^H	BS	Back Space	Ctrl-H
011	009	09	^I	HT	Horizontal Tab	Ctrl-I
012	010	0A	^J	LF	Line Feed	Ctrl-J
013	011	0B	^K	VT	Vertical Tab	Ctrl-K
014	012	0C	^L	FF	Form Feed	Ctrl-L
015	013	0D	^M	CR	Carriage Return	Ctrl-M
016	014	0E	^N	SO	Shift Out	Ctrl-N
017	015	0F	^O	SI	Shift In	Ctrl-O
020	016	10	^P	DLE	(or DEL) Delete	Ctrl-P
021	017	11	^Q	DC1	Device Control 1	Ctrl-Q
022	018	12	^R	DC2	Device Control 2	Ctrl-R
023	019	13	^S	DC3	Device Control 3	Ctrl-S
024	020	14	^T	DC4	Device Control 4	Ctrl-T
025	021	15	^U	NAK	Negative Acknowledge	Ctrl-U
026	022	16	^V	SYN	Synchronize	Ctrl-V
027	023	17	^W	ETB	End Transmission Block	Ctrl-W
030	024	18	^X	CAN	Cancel	Ctrl-X
031	025	19	^Y	EM	End of Medium	Ctrl-Y
032	026	1A	^Z	SUB	Substitute	Ctrl-Z
033	026	1A	^[ESC	Escape	Ctrl-[
034	026	1A	^\	FS	File Separator	Ctrl-\
035	026	1A	^]	GS	Group Separator	Ctrl-]
036	026	1A	^^	RS	Record Separator	Ctrl-Shift-^
037	026	1A	^_	US	Unit Separator	Ctrl-Shift-_
177	127	7F	^?	DEL	Delete	DEL

WARNINGS

Note that some HP-UX subsystems such as the keyboard interface, window systems, and other system software may use selected keyboard control characters for special purposes, possibly causing unexpected results.

FILES

`/usr/share/lib/pub/ascii`

a

NAME

Audio - audio tools available through HP VUE

DESCRIPTION

This man-page describes the audio tools available through **HP VUE** for playing, recording, and editing sound. These include Audio Setup, Audio Security, Audio Editor, Audio Control Panel, Audio File and Data Formats, and Audio Library. *Audio(5)* also provides information on using other audio tools from the HP-UX command line.

Audio Setup Requirements

To use the audio tools, you need access to both audio client and server software. This software is part of HP-UX. The server requires a workstation or X station with Audio hardware.

Audio hardware is built into all Series 700 computers *except* the 720, 730, and 750; you can upgrade these models to become the 725, 735, or 755, which do have audio hardware. Note, older 705s (that is, 705s with the 8MB HP-UX) do not include audio software.

To use audio on an X station, you need either an HP ENVIZEX or ENTRIA X station that includes an audio accessory kit.

In most cases, you use the audio client and server software on one system. However, if you need the audio server running on a remote workstation or X station, see *aserver(1M)*. The audio data files can reside on either system or a third system.

Audio Security

Audio is secured to allow access only to the user on the local workstation. If you need to allow remote systems to access audio on a workstation, see *asecure(1M)*.

The Audio Editor

The Audio Editor is an OSF/Motif-based tool with play, record, and edit functions. The Editor displays a waveform that makes it easy to edit and play audio segments.

You can open an audio file, play it, look at its waveform, and use the waveform controls to edit the file. To set an output device, use the Audio Control Panel.

To record audio, first connect a microphone or other audio equipment that your system supports; perhaps a CD or tape player. To make the connections, see the Audio Editor online help ("Audio Editor Tasks" section) or your system owner's manual.

You can use the Editor to create and record an audio file.

To start the Editor from the General Toolbox, open the Media Toolbox and drop an audio file on the Audio Editor control or double-click the control.

To start the Editor in a terminal window, type the following:

```
/opt/audio/bin/audio_editor [pathname]
```

Online help is available through the **Help** menu in the upper-right hand corner of the Editor.

The Audio Control Panel

The Audio Control Panel is an OSF/Motif-based tool that you use to set the audio volume and choose the audio device for playback.

The volume control affects the play volume for any client system of this workstation or X station. The Audio Control Panel also includes a Stop button to stop the current play operation.

You can also use the Audio Control Panel to choose the device for playback, (headphones, built-in speaker, or device connected to Line Out, such as external speakers). This choice controls where audio is played when you double-click an audio file or use the Audio Editor to play a file. The default output device is the internal (built-in) speaker.

To start the Audio Control Panel from HP VUE, click on the Audio control in HP VUE Front Panel.

To start the Audio Control Panel in a terminal window, type the following:

```
/opt/audio/bin/AudioCP
```

If your system has Audio applications that were developed using an earlier version of Audio software, those applications may use the **SPEAKER** environment variable to determine their output devices. You can set

the **SPEAKER** variable for all applications started by HP VUE by modifying the `$HOME/.vueprofile` file. The **SPEAKER** variable can be external (headphones, Line Out) or internal (built-in speaker).

To set the **SPEAKER** variable for a Bourne or Korn shell, enter:

```
SPEAKER=internal
export speaker
```

To set the **SPEAKER** variable for a C shell, enter:

```
setenv SPEAKER internal
```

Audio File and Data Formats

Audio files supported contain uncompressed audio data in one of three file formats: generic, RIFF/Waveform, or raw. Each file also needs the correct filename extension. For the three file formats, the Audio Editor online help lists which data formats and filename extensions that apply.

The extension causes the appropriate icon to appear in the File Manager. To play audio files you can drag and drop the file icons onto the Audio Editor or Control Panel or double-click the icon.

If you need to add an extension to a filename, (or convert the file format) the `/opt/audio/bin/convert` command is recommended. See `convert(1)`. However, you can instead rename the file to make it playable. Use this filename format:

filename.rate.data_type

The *rate* and *data_type* variables accept values defined for the `-drate` and `-ddata` options of `convert`. If needed, you can omit the *rate* variable. Use this filename format:

filename.data_type

Audio Library

HP-UX includes an Audio Library that was used to build the audio tools. If you have ordered and installed the User Environment Developer's Kit, you can use the Audio Library to create other audio applications.

The HP-UX Audio Library contains functions that C programs can use to manipulate audio. The functions interact with the Audio Server, enabling the application to record and play audio data files and convert audio data files from one format to another.

For more information about audio programming, refer to the manual *Using the Audio Developer's Kit*.

AUTHOR

The Audio Library, Audio Editor, and Audio Control Panel were developed by HP.

SEE ALSO

`asecure(1M)`, `aserver(1M)`, `attributes(1)`, `convert(1)`, `send_sound(1)`.

Using the Audio Developer's Kit

NAME

audit - introduction to HP-UX Auditing System

SYNOPSIS

```
#include <sys/audit.h>
```

DESCRIPTION

The purpose of the auditing system is to record instances of access by subjects to objects and to allow detection of any (repeated) attempts to bypass the protection mechanism and any misuses of privileges, thus acting as a deterrent against system abuses and exposing potential security weaknesses in the system.

User and Event Selection

The auditing system provides administrators with a mechanism to select users and activities to be audited. Users are assigned unique identifiers called **audit ids** by the administrator which remain unchanged throughout a user's history. The *audusr*(1M) command is used to specify those users who are to be audited. The *audevent*(1M) command is used to specify system activities (auditable events) that are to be audited. Auditable events are classified into several categories, illustrated by the event category list at the end. (An event category consists of a set of operations that affect a particular aspect of the system.)

Self-auditing Programs

To reduce the amount of log data and to provide a higher-level recording of some typical system operations, a collection of privileged programs are given capabilities to perform self-auditing. This means that the programs can suspend the currently specified auditing on themselves and produce a high-level description of the operations they perform. These self-auditing programs include: *at*(1), *chfn*(1), *chsh*(1), *crontab*(1), *login*(1), *newgrp*(1), *passwd*(1), *audevent*(1M), *audisp*(1M), *audsys*(1M), *audusr*(1M), *cron*(1M), *init*(1M), *lpsched*(1M), *pwck*(1M), and *sam*(1M). Note that only these privileged programs are allowed to do self-auditing, and that the audit suspension they perform only affects these programs and does not affect any other processes on the system.

Viewing of Audited Data

The *audisp*(1M) command is used to view audited data recorded in log file(s). *audisp*(1M) merges the log file(s) into a single audit trail in chronological sequence. The administrator can select viewing criteria provided by *audisp*(1M) to limit the search to particular kinds of events which the administrator is interested in investigating.

Monitoring the Auditing System

To ensure that the auditing system operates normally and that any abnormal behaviors are detected, a privileged **daemon** program, *audomon*(1M), runs in the background to monitor various auditing system parameters. When these parameters take on abnormal (dangerous) values, or when components of the auditing system are accidentally removed, *audomon*(1M) prints warning messages and tries to resolve the problem if possible.

Starting and Halting the Auditing System

The administrator can use the *audsys*(1M) command to start or halt the auditing system, or to get a brief summary of the status of the audit system. Prior to starting the auditing system, *audsys*(1M) also validates the parameters specified, and ensures that the auditing system is in a safe and consistent state.

Audit Log Files

At any time when the auditing system is enabled, at least an audit log file must be present, and another back-up log file is highly recommended. Both of these files (along with various attributes for these files) can be specified using *audsys*(1M). When the current log file exceeds a pre-specified size, or when the auditing file system is dangerously full, the system automatically switches to the back-up file if possible. If a back-up log file is not available, warning messages are sent to request appropriate administrator action.

Event Categories

create	Log creations of objects (files, directories, other objects), including <i>creat</i> (2), <i>mkdir</i> (2), <i>mknod</i> (2), <i>msgget</i> (2), <i>pipe</i> (2), <i>semget</i> (2), <i>shmat</i> (2), and <i>shmget</i> (2).
delete	Log all deletions of objects (files, directories, other objects), including <i>ksem_unlink</i> (2), <i>mq_unlink</i> (2), <i>msgctl</i> (2), <i>rmdir</i> (2), <i>semctl</i> (2), and <i>shm_unlink</i> (2).
readdac	Log reads of Discretionary access control (DAC) information including <i>access</i> (2), <i>fstat</i> (2), <i>fstat64</i> (2), <i>getaccess</i> (2), <i>lstat</i> (2), <i>lstat64</i> (2), <i>stat</i> (2), and <i>stat64</i> (2).

moddac	Log all modifications of object's Discretionary access control (DAC) information including <i>chmod(2)</i> , <i>chown(2)</i> , <i>fchmod(2)</i> , <i>fchown(2)</i> , <i>fsetacl(2)</i> , <i>setacl(2)</i> , and <i>umask(2)</i> .
modaccess	Log all modifications other than DAC, including <i>chdir(2)</i> , <i>chroot(2)</i> , <i>link(2)</i> , <i>lockf64(2)</i> , <i>newgrp(1)</i> , <i>rename(2)</i> , <i>setgid(2)</i> , <i>setgroups(2)</i> , <i>setresgid(2)</i> , <i>setresuid(2)</i> , <i>setuid(2)</i> , <i>shmctl(2)</i> , <i>shmdt(2)</i> , and <i>unlink(2)</i> .
open	Log all openings of objects (file open, other objects' open) including <i>execv(2)</i> , <i>execve(2)</i> , <i>ftruncate(2)</i> , <i>ftruncate64(2)</i> , <i>kload(2)</i> , <i>ksem_open(2)</i> , <i>lpsched(1M)</i> , <i>mmap64(2)</i> , <i>mq_open(2)</i> , <i>open(2)</i> , <i>ptrace(2)</i> , <i>shm_open(2)</i> , <i>truncate(2)</i> , and <i>truncate64(2)</i> .
close	Log all closings of objects (file close, other objects' close) including <i>close(2)</i> , <i>ksem_close(2)</i> , and <i>mq_close(2)</i> .
process	Log all operations on processes, including <i>exit(2)</i> , <i>fork(2)</i> , <i>kill(2)</i> , <i>mlock(2)</i> , <i>mlockall(2)</i> , <i>munlock(2)</i> , <i>munlockall(2)</i> , <i>setcontext(2)</i> , <i>setrlimit64(2)</i> , <i>sigqueue(2)</i> , <i>ulimit64(2)</i> , and <i>vfork(2)</i> .
removable	Log all removable media events (mounting and unmounting events), including <i>mount(2)</i> , <i>umount(2)</i> , and <i>vfsmount(2)</i> .
login	Log all logins and logouts, including <i>login(1)</i> , <i>init(1M)</i> .
admin	Log all administrative and privileged events, including <i>audevent(1M)</i> , <i>audisp(1M)</i> , <i>audswitch(2)</i> , <i>audsys(1M)</i> , <i>audusr(1M)</i> , <i>chfn(1)</i> , <i>chsh(1)</i> , <i>init(1M)</i> , <i>passwd(1)</i> , <i>pwck(1M)</i> , <i>reboot(2)</i> , <i>sam(1M)</i> , <i>setaudit(2)</i> , <i>setauditproc(2)</i> , <i>setdomainname(2)</i> , <i>setevent(2)</i> , <i>sethostid(2)</i> , <i>settimeofday(2)</i> , <i>stime(2)</i> , and <i>swapon(2)</i> .
ipccreat	Log all IPC create events including <i>socket(2)</i> and <i>bind(2)</i> .
ipcopen	Log all IPC open events including <i>connect(2)</i> and <i>accept(2)</i> .
ipcclose	Log all IPC close events including <i>shutdown(2)</i> .
uevent1	Log user-defined event.
uevent2	Log user-defined event.
uevent3	Log user-defined event.
ipcdgram	Log IPC Datagram transactions.

For a complete description of system call assignments to event types, see *audevent(1M)*.

Note that some commands such as *init(1M)* may occur in more than one category because the event varies, depending on the operation done by the command.

AUTHOR

The auditing system described above was developed by HP.

SEE ALSO

audsys(1M), *audusr(1M)*, *audevent(1M)*, *audisp(1M)*, *audctl(2)*, *audswitch(2)*, *audwrite(2)*, *getaudit(2)*, *setaudit(2)*, *getevent(2)*, *setevent(2)*, *audit(4)*.

NAME

curses.h — definitions for screen handling and optimisation functions

SYNOPSIS

```
#include <curses.h>
```

DESCRIPTION**Objects**

The `<curses.h>` header provides a declaration for *COLOR_PAIRS*, *COLORS*, *COLS*, *curscr*, *LINES* and *stdscr*.

Constants

The following constants are defined:

EOF	Function return value for end-of-file
ERR	Function return value for failure
FALSE	Boolean <i>false</i> value
OK	Function return value for success
TRUE	Boolean <i>true</i> value
WEOF	Wide-character function return value for end-of-file, as defined in <code><wchar.h></code> .

Data Types

The following data types are defined through **typedef**:

attr_t	An OR-ed set of attributes
bool	Boolean data type
chtype	A character, attributes and a colour-pair
SCREEN	An opaque terminal representation
wchar_t	As described in <code><stddef.h></code>
wint_t	As described in <code><wchar.h></code>
cchar_t	References a string of wide characters
WINDOW	An opaque window representation

These data types are described in more detail in *Data Types* in `curses_intro`.

The inclusion of `<curses.h>` may make visible all symbols from the headers `<stdio.h>`, `<term.h>`, `<termios.h>` and `<wchar.h>`.

Attribute Bits

The following symbolic constants are used to manipulate objects of type **attr_t**:

WA_ALTCHARSET	Alternate character set
WA_BLINK	Blinking
WA_BOLD	Extra bright or bold
WA_DIM	Half bright
WA_HORIZONTAL	Horizontal highlight
WA_INVIS	Invisible
WA_LEFT	Left highlight
WA_LOW	Low highlight
WA_PROTECT	Protected
WA_REVERSE	Reverse video
WA_RIGHT	Right highlight
WA_STANDOUT	Best highlighting mode of the terminal
WA_TOP	Top highlight
WA_UNDERLINE	Underlining
WA_VERTICAL	Vertical highlight

These attribute flags shall be distinct.

(CURSES)

The following symbolic constants are used to manipulate attribute bits in objects of type **chtype**:

A_ALTCHARSET	Alternate character set
A_BLINK	Blinking
A_BOLD	Extra bright or bold
A_DIM	Half bright
A_INVIS	Invisible
A_PROTECT	Protected
A_REVERSE	Reverse video
A_STANDOUT	Best highlighting mode of the terminal
A_UNDERLINE	Underlining

These attribute flags need not be distinct except when `_XOPEN_CURSES` is defined and the application sets `_XOPEN_SOURCE_EXTENDED` to 1.

The following symbolic constants can be used as bit-masks to extract the components of a **chtype**:

A_ATTRIBUTES	Bit-mask to extract attributes
A_CHARTEXT	Bit-mask to extract a character
A_COLOR	Bit-mask to extract colour-pair information

The following symbolic constants can be used as bit-masks to extract the components of a **chtype**:

A_ATTRIBUTES	Bit-mask to extract attributes
A_CHARTEXT	Bit-mask to extract a character
A_COLOR	Bit-mask to extract colour-pair information

Line-Drawing Constants

The `<curses.h>` header defines the symbolic constants shown in the leftmost two columns of the following table for use in drawing lines. The symbolic constants that begin with `ACS_` are **char** constants. The symbolic constants that begin with `WACS_` are **cchar_t** constants for use with the wide-character interfaces that take a pointer to a **cchar_t**.

In the POSIX locale, the characters shown in the *POSIX Locale Default* column are used when the terminal database does not specify a value using the **acsc** capability as described in *Line Graphics* in *terminfo(4)*.

POSIX Locale			
char Constant	cchar_t Constant	Default	Glyph Description
ACS_ULCORNER	WACS_ULCORNER	+	upper left-hand corner
ACS_LLCORNER	WACS_LLCORNER	+	lower left-hand corner
ACS_URCORNER	WACS_URCORNER	+	upper right-hand corner
ACS_LRCORNER	WACS_LRCORNER	+	lower right-hand corner
ACS_RTEE	WACS_RTEE	+	right tee (⌋)
ACS_LTEE	WACS_LTEE	+	left tee (⌈)
ACS_BTEE	WACS_BTEE	+	bottom tee (⌞)
ACS_TTEE	WACS_TTEE	+	top tee (⌟)
ACS_HLINE	WACS_HLINE	—	horizontal line
ACS_VLINE	WACS_VLINE		vertical line
ACS_PLUS	WACS_PLUS	+	plus
ACS_S1	WACS_S1	—	scan line 1
ACS_S9	WACS_S9	—	scan line 9
ACS_DIAMOND	WACS_DIAMOND	+	diamond
ACS_CKBOARD	WACS_CKBOARD	:	checker board (stipple)
ACS_DEGREE	WACS_DEGREE	'	degree symbol
ACS_PLMINUS	WACS_PLMINUS	#	plus/minus
ACS_BULLET	WACS_BULLET	o	bullet
ACS_LARROW	WACS_LARROW	<	arrow pointing left
ACS_RARROW	WACS_RARROW	>	arrow pointing right
ACS_DARROW	WACS_DARROW	v	arrow pointing down
ACS_UARROW	WACS_UARROW	^	arrow pointing up
ACS_BOARD	WACS_BOARD	#	board of squares
ACS_LANTERN	WACS_LANTERN	#	lantern symbol
ACS_BLOCK	WACS_BLOCK	#	solid square block

(CURSES)

Colour-related Macros

The following colour-related macros are defined:

```
COLOR_BLACK
COLOR_BLUE
COLOR_GREEN
COLOR_CYAN
COLOR_RED
COLOR_MAGENTA
COLOR_YELLOW
COLOR_WHITE
```

Coordinate-related Macros

The following coordinate-related macros are defined:

```
void    getbegyx(WINDOW *win, int y, int x);
void    getmaxyx(WINDOW *win, int y, int x);
void    getparyx(WINDOW *win, int y, int x);
void    getyx(WINDOW *win, int y, int x);
```

(CURSES)

Key Codes

The following symbolic constants representing function key values are defined:

Key Code	Description
KEY_CODE_YES	Used to indicate that a <code>wchar_t</code> variable contains a key code
KEY_BREAK	Break key
KEY_DOWN	Down arrow key
KEY_UP	Up arrow key
KEY_LEFT	Left arrow key
KEY_RIGHT	Right arrow key
KEY_HOME	Home key
KEY_BACKSPACE	Backspace
KEY_F0	Function keys; space for 64 keys is reserved
KEY_F(<i>n</i>)	For $0 \leq n \leq 63$
KEY_DL	Delete line
KEY_IL	Insert line
KEY_DC	Delete character
KEY_IC	Insert char or enter insert mode
KEY_EIC	Exit insert char mode
KEY_CLEAR	Clear screen
KEY_EOS	Clear to end of screen
KEY_EOL	Clear to end of line
KEY_SF	Scroll 1 line forward
KEY_SR	Scroll 1 line backward (reverse)
KEY_NPAGE	Next page
KEY_PPAGE	Previous page
KEY_STAB	Set tab
KEY_CTAB	Clear tab
KEY_CATAB	Clear all tabs
KEY_ENTER	Enter or send
KEY_SRESET	Soft (partial) reset
KEY_RESET	Reset or hard reset
KEY_PRINT	Print or copy
KEY_LL	Home down or bottom
KEY_A1	Upper left of keypad
KEY_A3	Upper right of keypad
KEY_B2	Center of keypad
KEY_C1	Lower left of keypad
KEY_C3	Lower right of keypad

The virtual keypad is a 3-by-3 keypad arranged as follows:

A1	UP	A3
LEFT	B2	RIGHT
C1	DOWN	C3

Each legend, such as A1, corresponds to a symbolic constant for a key code from the preceding table, such as KEY_A1. The following symbolic constants representing function key values are also defined:

Key Code	Description
KEY_BTAB	Back tab key
KEY_BEG	Beginning key
KEY_CANCEL	Cancel key
KEY_CLOSE	Close key
KEY_COMMAND	Cmd (command) key
KEY_COPY	Copy key
KEY_CREATE	Create key
KEY_END	End key
KEY_EXIT	Exit key
KEY_FIND	Find key
KEY_HELP	Help key
KEY_MARK	Mark key
KEY_MESSAGE	Message key
KEY_MOVE	Move key

(CURSES)

KEY_NEXT	Next object key
KEY_OPEN	Open key
KEY_OPTIONS	Options key
KEY_PREVIOUS	Previous object key
KEY_REDO	Redo key
KEY_REFERENCE	Reference key
KEY_REFRESH	Refresh key
KEY_REPLACE	Replace key
KEY_RESTART	Restart key
KEY_RESUME	Resume key
KEY_SAVE	Save key
KEY_SBEG	Shifted beginning key
KEY_SCANCEL	Shifted cancel key
KEY_SCOMMAND	Shifted command key
KEY_SCOPY	Shifted copy key
KEY_SCREATE	Shifted create key
KEY_SDC	Shifted delete char key
KEY_SDL	Shifted delete line key
KEY_SELECT	Select key
KEY_SEND	Shifted end key
KEY_SEOL	Shifted clear line key
KEY_SEXIT	Shifted exit key
KEY_SFIND	Shifted find key
KEY_SHELP	Shifted help key
KEY_SHOME	Shifted home key
KEY_SIC	Shifted input key
KEY_SLEFT	Shifted left arrow key
KEY_SMESSAGE	Shifted message key
KEY_SMOVE	Shifted move key
KEY_SNEXT	Shifted next key
KEY_SOPTIONS	Shifted options key
KEY_SPREVIOUS	Shifted prev key
KEY_SPRINT	Shifted print key
KEY_SREDO	Shifted redo key
KEY_SREPLACE	Shifted replace key
KEY_SRIGHT	Shifted right arrow
KEY_SRSUME	Shifted resume key
KEY_SSAVE	Shifted save key
KEY_SSUSPEND	Shifted suspend key
KEY_SUNDO	Shifted undo key
KEY_SUSPEND	Suspend key
KEY_UNDO	Undo key

Function Prototypes

The following are declared as functions, and may also be defined as macros:

```

int      addch(const chtype ch);
int      addchstr(const chtype *chstr);
int      addchnstr(const chtype *chstr, int n);
int      addnstr(const char *str, int n);
int      addstr(const char *str);
int      addnwstr(const wchar_t *wstr, int n);
int      addwstr(const wchar_t *wstr);
int      add_wch(const cchar_t *wch);
int      add_wchnstr(const cchar_t *wchstr, int n);
int      add_wchstr(const cchar_t *wchstr);
int      attroff(int attrs);
int      attron(int attrs);
int      attrset(int attrs);
int      attr_get(attr_t *attrs, short *color_pair, void *opts);
int      attr_off(attr_t attrs, void *opts);
int      attr_on(attr_t attrs, void *opts);
int      attr_set(attr_t attrs, short color_pair, void *opts);

```



```

int     baudrate(void);
int     beep(void);
int     bkgd(chtype ch);
void    bkgdset(chtype ch);
void    bkgrndset(const cchar_t *wch);
int     bkgrnd(const cchar_t *wch);
int     border(chtype ls, chtype rs, chtype ts, chtype tl,
               chtype tr, chtype bl, chtype br);
int     border_set(const cchar_t *ls, const cchar_t *rs,
                  const cchar_t *ts, const cchar_t *bs,
                  const cchar_t *tl, const cchar_t *tr,
                  const cchar_t *bl, const cchar_t *br);
int     box(WINDOW *win, chtype verch, chtype horch);
int     box_set(WINDOW *win, const cchar_t *verch,
               const cchar_t *horch);
bool    can_change_color(void);
int     cbreak(void);
int     chgat(int n, attr_t attr, short color, const void *opts);
int     clear(void);
int     clearok(WINDOW *win, bool bf);
int     clrtoebot(void);
int     clrtoeol(void);
int     color_content(short color, short *red, short *green,
                    short *blue);
int     COLOR_PAIR(int n);
int     copywin(const WINDOW *srcwin, WINDOW *dstwin, int sminrow,
               int smincol, int dminrow, int dmincol, int dmaxrow,
               int dmaxcol, int overlay);
int     curs_set(int visibility);
int     def_prog_mode(void);
int     def_shell_mode(void);
int     delay_output(int ms);
int     delch(void);
void    delscreen(SCREEN *sp);
int     delwin(WINDOW *win);
int     deleteln(void);
WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y,
               int begin_x);
int     doupdate(void);
WINDOW *dupwin(WINDOW *win);
int     echo(void);
int     echochar(const chtype ch);
int     echo_wchar(const cchar_t *wch);
int     endwin(void);
int     erase(void);
char    erasechar(void);
int     erasewchar(wchar_t *ch);
void    filter(void);
int     flash(void);
int     flushinp(void);
chtype  getbkgd(WINDOW *win);
int     getbkgrnd(cchar_t *wch);
int     getcchar(const cchar_t *wcval, wchar_t *wch, attr_t *attrs,
               short *color_pair, void *opts);
int     getch(void);
int     getnstr(char *str, int n);
int     getn_wstr(wint_t *wstr, int n);
int     getstr(char *str);
int     get_wch(wint_t *ch);
WINDOW *getwin(FILE *filep);
int     get_wstr(wint_t *wstr);
int     halfdelay(int tenths);

```

C

(CURSES)

```

bool    has_colors(void);
bool    has_ic(void);
bool    has_il(void);
int     hline(chtype ch, int n);
int     hline_set(const cchar_t *wch, int n);
void    idlok(WINDOW *win, bool bf);
int     idlok(WINDOW *win, bool bf);
void    immedok(WINDOW *win, bool bf);
chtype  inch(void);
int     inchnstr(chtype *chstr, int n);
int     inchstr(chtype *chstr);
WINDOW  *initscr(void);
int     init_color(short color, short red, short green, short blue);
int     init_pair(short pair, short f, short b);
int     innstr(char *str, int n);
int     innwstr(wchar_t *wstr, int n);
int     insch(chtype ch);
int     insdelln(int n);
int     insertln(void);
int     insnstr(const char *str, int n);
int     insstr(const char *str);
int     instr(char *str);
int     ins_nwstr(const wchar_t *wstr, int n);
int     ins_wch(const cchar_t *wch);
int     ins_wstr(const wchar_t *wstr);
int     intrflush(WINDOW *win, bool bf);
int     in_wch(cchar_t *wcval);
int     in_wchstr(cchar_t *wchstr);
int     in_wchnstr(cchar_t *wchstr, int n);
int     inwstr(wchar_t *wstr);
bool    isendwin(void);
bool    is_linetouched(WINDOW *win, int line);
bool    is_wintouched(WINDOW *win);
char    *keyname(int c);
char    *key_name(wchar_t c);
int     keypad(WINDOW *win, bool bf);
char    killchar(void);
int     killwchar(wchar_t *ch);
int     leaveok(WINDOW *win, bool bf);
char    *longname(void);
int     meta(WINDOW *win, bool bf);
int     move(int y, int x);
int     mvaddch(int y, int x, const chtype ch);
int     mvaddchnstr(int y, int x, const chtype *chstr, int n);
int     mvaddchstr(int y, int x, const chtype *chstr);
int     mvaddnstr(int y, int x, const char *str, int n);
int     mvaddnwstr(int y, int x, const wchar_t *wstr, int n);
int     mvaddstr(int y, int x, const char *str);
int     mvaddwstr(int y, int x, const wchar_t *wstr);
int     mvadd_wch(int y, int x, const cchar_t *wch);
int     mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n);
int     mvadd_wchstr(int y, int x, const cchar_t *wchstr);
int     mvchgat(int y, int x, int n, attr_t attr, short color,
               const void *opts);
int     mvcur(int oldrow, int oldcol, int newrow, int newcol);
int     mvdelch(int y, int x);
int     mvderwin(WINDOW *win, int par_y, int par_x);
int     mvgetch(int y, int x);

```

(CURSES)

```

int     mvgetnstr(int y, int x, char *str, int n);
int     mvgetn_wstr(int y, int x, wint_t *wstr, int n);
int     mvgetstr(int y, int x, char *str);
int     mvget_wch(int y, int x, wint_t *ch);
int     mvget_wstr(int y, int x, wint_t *wstr);
int     mvhline(int y, int x, chtype ch, int n);
int     mvhline_set(int y, int x, const cchar_t *wch, int n);
chtype mvinch(int y, int x);
int     mvinchnstr(int y, int x, chtype *chstr, int n);
int     mvinchstr(int y, int x, chtype *chstr);
int     mvinnstr(int y, int x, char *str, int n);
int     mvinnwstr(int y, int x, wchar_t *wstr, int n);
int     mvinsch(int y, int x, chtype ch);
int     mvinsnstr(int y, int x, const char *str, int n);
int     mvinsstr(int y, int x, const char *str);
int     mvinstr(int y, int x, char *str);
int     mvins_wstr(int y, int x, const wchar_t *wstr, int n);
int     mvins_wch(int y, int x, const cchar_t *wch);
int     mvins_wstr(int y, int x, const wchar_t *wstr);
int     mvinwstr(int y, int x, wchar_t *wstr);
int     mvin_wch(int y, int x, cchar_t *wchval);
int     mvin_wchnstr(int y, int x, cchar_t *wchstr, int n);
int     mvin_wchstr(int y, int x, cchar_t *wchstr);
int     mvprintw(int y, int x, char *fmt, ...);
int     mvscanw(int y, int x, char *fmt, ...);
int     mvvline(int y, int x, chtype ch, int n);
int     mvvline_set(int y, int x, const cchar_t *wch, int n);
int     mvwaddch(WINDOW *win, int y, int x, const chtype ch);
int     mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr,
                    int n);
int     mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr);
int     mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n);
int     mvwaddnwstr(WINDOW *win, int y, int x, const wchar_t *wstr,
                    int n);
int     mvwaddstr(WINDOW *win, int y, int x, const char *str);
int     mvwaddwstr(WINDOW *win, int y, int x, const wchar_t *wstr);
int     mvwadd_wch(WINDOW *win, int y, int x, const cchar_t *wch);
int     mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr,
                    int n);
int     mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr);
int     mvwchgat(WINDOW *win, int y, int x, int n, attr_t attr,
                short color, const void *opts);
int     mvwdelch(WINDOW *win, int y, int x);
int     mvwgetch(WINDOW *win, int y, int x);
int     mvwgetnstr(WINDOW *win, int y, int x, char *str, int n);
int     mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n);
int     mvwgetstr(WINDOW *win, int y, int x, char *str);
int     mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);
int     mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr);
int     mvwhline(WINDOW *win, int y, int x, chtype ch, int n);
int     mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch,
                    int n);
int     mvwin(WINDOW *win, int y, int x);
chtype mvwinch(WINDOW *win, int y, int x);
int     mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);
int     mvwinchstr(WINDOW *win, int y, int x, chtype *chstr);
int     mvwinnstr(WINDOW *win, int y, int x, char *str, int n);
int     mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);

```

C

(CURSES)

```

int      mvwinsch(WINDOW *win, int y, int x, chtype ch);
int      mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n);
int      mvwinsstr(WINDOW *win, int y, int x, const char *str);
int      mvwinstr(WINDOW *win, int y, int x, char *str);
int      mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr,
                     int n);
int      mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch);
int      mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr);
int      mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int      mvwin_wch(WINDOW *win, int y, int x, cchar_t *wcval);
int      mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr,
                     int n);
int      mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr);
int      mvwprintw(WINDOW *win, int y, int x, char *fmt, ...);
int      mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int      mvwvline(WINDOW *win, int y, int x, chtype ch, int n);
int      mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch,
                     int n);
int      napms(int ms);
WINDOW   *newpad(int nlines, int ncols);
SCREEN   *newterm(char *type, FILE *outfile, FILE *infile);
WINDOW   *newwin(int nlines, int ncols, int begin_y, int begin_x);
int      nl(void);
int      nonl(void);
int      nocbreak(void);
int      nodelay(WINDOW *win, bool bf);
int      noecho(void);
void     noqiflush(void);
int      noraw(void);
int      notimeout(WINDOW *win, bool bf);
int      overlay(const WINDOW *srcwin, WINDOW *dstwin);
int      overwrite(const WINDOW *srcwin, WINDOW *dstwin);
int      pair_content(short pair, short *f, short *b);
int      PAIR_NUMBER(int value);
int      pechochar(WINDOW *pad, chtype *ch);
int      pecho_wchar(WINDOW *pad, const cchar_t *wch);
int      pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow,
                     int smincol, int smaxrow, int smaxcol);
int      prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow,
                  int smincol, int smaxrow, int smaxcol);
int      printw(char *fmt, ...);
int      putp(const char *str);
int      putwin(WINDOW *win, FILE *filep);
void     qiflush(void);
int      raw(void);
int      redrawwin(WINDOW *win);
int      refresh(void);
int      resetty(void);
int      reset_prog_mode(void);
int      reset_shell_mode(void);
int      ripoffline(int line, int (*init)(WINDOW *win, int columns));
int      savetty(void);
int      scanw(char *fmt, ...);
int      scr_dump(const char *filename);
int      scr_init(const char *filename);
int      scrll(int n);
int      scroll(WINDOW *win);
int      scrollok(WINDOW *win, bool bf);
int      scr_restore(const char *filename);
int      scr_set(const char *filename);
int      setcchar(cchar_t *wcval, const wchar_t *wch,
                 const attr_t attrs, short color_pair,

```

```

        const void *opts);
int     setscrreg(int top, int bot);
SCREEN *set_term(SCREEN *new);
int     slk_attron(const chtype attrs);
int     slk_attr_off(const attr_t attrs, void *opts);
int     slk_attron(const attr_t attrs, void *opts);
int     slk_attrset(const chtype attrs);
int     slk_attr_set(const attr_t attrs, short color_pair,
                    void *opts);
int     slk_clear(void);
int     slk_init(int fmt);
char *slk_label(int labnum);
int     slk_noutrefresh(void);
int     slk_refresh(void);
int     slk_restore(void);
int     slk_set(int labnum, const char *label, int justify);
int     slk_touch(void);
int     slk_wset(int labnum, const wchar_t *label, int justify);
int     standend(void);
int     standout(void);
int     start_color(void);
WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y,
               int begin_x);
WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y,
               int begin_x);
int     syncok(WINDOW *win, bool bf);
chtype termattrs(void);
attr_t term_attrs(void);
char *termname(void);
int     tigetflag(char *capname);
int     tigetnum(char *capname);
char *tigetstr(char *capname);
void    timeout(int delay);
int     touchline(WINDOW *win, int start, int count);
int     touchwin(WINDOW *win);
char *tparm(char *cap, long p1, long p2, long p3, long p4,
             long p5, long p6, long p7, long p8, long p9);
int     typeahead(int fildes);
int     ungetch(int ch);
int     unget_wch(const wchar_t wch);
int     untouchwin(WINDOW *win);
void    use_env(bool boolvalue);
int     vidattr(chtype attr);
int     vid_attr(attr_t attr, short color_pair, void *opts);
int     vidputs(chtype attr, int (*putfunc)(int));
int     vid_puts(attr_t attr, short color_pair, void *opt,
                 int (*putwfunc)(int));
int     vline(chtype ch, int n);
int     vline_set(const cchar_t *wch, int n);
int     vwprintw(WINDOW *win, char *fmt, void *varglist);
int     vw_printw(WINDOW *win, char *fmt, void *varglist);
int     vwscanw(WINDOW *win, char *fmt, void *varglist);
int     vw_scanw(WINDOW *win, char *fmt, void *varglist);
int     waddch(WINDOW *win, const chtype ch);
int     waddchnstr(WINDOW *win, const chtype *chstr, int n);
int     waddchstr(WINDOW *win, const chtype *chstr);
int     waddnstr(WINDOW *win, const char *str, int n);
int     waddnwstr(WINDOW *win, const wchar_t *wstr, int n);
int     waddstr(WINDOW *win, const char *str);
int     waddwstr(WINDOW *win, const wchar_t *wstr);
int     wadd_wch(WINDOW *win, const cchar_t *wch);

```

C

(CURSES)

```

int      wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);
int      wadd_wchstr(WINDOW *win, const cchar_t *wchstr);
int      wattroff(WINDOW *win, int attrs);
int      wattron(WINDOW *win, int attrs);
int      wattrset(WINDOW *win, int attrs);
int      wattr_get(WINDOW *win, attr_t *attrs, short *color_pair, void *opts);
int      wattr_off(WINDOW *win, attr_t attrs, void *opts);
int      wattr_on(WINDOW *win, attr_t attrs, void *opts);
int      wattr_set(WINDOW *win, attr_t attrs, void *opts);
int      wbkgd(WINDOW *win, chtype ch);
void     wbkgdset(WINDOW *win, chtype ch);
int      wbkggrnd(WINDOW *win, const cchar_t *wch);
void     wbkggrndset(WINDOW *win, const cchar_t *wch);
int      wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs,
                 chtype tl, chtype tr, chtype bl, chtype br);
int      wborder_set(WINDOW *win, const cchar_t *ls, const cchar_t *rs,
                    const cchar_t *ts, const cchar_t *bs,
                    const cchar_t *tl, const cchar_t *tr,
                    const cchar_t *bl, const cchar_t *br);
int      wchgat(WINDOW *win, int n, attr_t attr, short color,
                const void *opts);
int      wclear(WINDOW *win);
int      wclrtoebot(WINDOW *win);
int      wclrtoeol(WINDOW *win);
void     wcursyncup(WINDOW *win);
int      wdelch(WINDOW *win);
int      wdeleteln(WINDOW *win);
int      wechochar(WINDOW *win, const chtype ch);
int      wecho_wchar(WINDOW *win, const cchar_t *wch);
int      werase(WINDOW *win);
int      wgetbkgrnd(WINDOW *win, cchar_t *wch);
int      wgetch(WINDOW *win);
int      wgetnstr(WINDOW *win, char *str, int n);
int      wgetn_wstr(WINDOW *win, wint_t *wstr, int n);
int      wgetstr(WINDOW *win, char *str);
int      wget_wch(WINDOW *win, wint_t *ch);
int      wget_wstr(WINDOW *win, wint_t *wstr);
int      whline(WINDOW *win, chtype ch, int n);
int      whline_set(WINDOW *win, const cchar_t *wch, int n);
chtype   winch(WINDOW *win);
int      winchnstr(WINDOW *win, chtype *chstr, int n);
int      winchstr(WINDOW *win, chtype *chstr);
int      winnstr(WINDOW *win, char *str, int n);
int      winnwstr(WINDOW *win, wchar_t *wstr, int n);
int      winsch(WINDOW *win, chtype ch);
int      winsdelln(WINDOW *win, int n);
int      winsertln(WINDOW *win);
int      winsnstr(WINDOW *win, const char *str, int n);
int      winsstr(WINDOW *win, const char *str);
int      winstr(WINDOW *win, char *str);
int      wins_nwstr(WINDOW *win, const wchar_t *wstr, int n);
int      wins_wch(WINDOW *win, const cchar_t *wch);
int      wins_wstr(WINDOW *win, const wchar_t *wstr);
int      winwstr(WINDOW *win, wchar_t *wstr);
int      win_wch(WINDOW *win, cchar_t *wcval);
int      win_wchnstr(WINDOW *win, cchar_t *wchstr, int n);
int      win_wchstr(WINDOW *win, cchar_t *wchstr);
int      wmove(WINDOW *win, int y, int x);
int      wnoutrefresh(WINDOW *win);
int      wprintw(WINDOW *win, char *fmt, ...);
int      wredrawln(WINDOW *win, int beg_line, int num_lines);
int      wrefresh(WINDOW *win);

```

(CURSES)

```

int      wscanw(WINDOW *win, char *fmt, ...);
int      wscr1(WINDOW *win, int n);
int      wsetscrreg(WINDOW *win, int top, int bot);
int      wstandend(WINDOW *win);
int      wstandout(WINDOW *win);
void     wsyncdown(WINDOW *win);
void     wsyncup(WINDOW *win);
void     wtimeout(WINDOW *win, int delay);
int      wtouchln(WINDOW *win, int y, int n, int changed);
wchar_t  *wunctrl(cchar_t *wc);
int      wvline(WINDOW *win, chtype ch, int n);
int      wvline_set(WINDOW *win, const cchar_t *wch, int n);

```

APPLICATION USAGE

In order to support historical applications that include `<curses.h>` and use `<varargs.h>` the following interfaces using `va_list` are declared as having a third argument of type `(void *)`: `vw_printw()`, `vw_scanw()`, `vwprintw()`, `vwscanw()`.

SEE ALSO

`<stdio.h>` (in the **X/Open System Interfaces and Headers, Issue 4, Version 2** specification), `<term.h>`, `<termios.h>` (in the **X/Open System Interfaces and Headers, Issue 4, Version 2** specification), `<unctrl.h>`, `<wchar.h>` (in the **X/Open System Interfaces and Headers, Issue 4, Version 2** specification).

`curses_intro(3X)`.

CHANGE HISTORY

First released in X/Open Curses, Issue 2.

X/Open Curses, Issue 4

The entry is completely rewritten to include new constants, data types and function prototypes.

NAME

dirent.h - format of directory streams and directory entries

SYNOPSIS

```
#include <sys/types.h>
#include <dirent.h>
```

DESCRIPTION

This header file defines data types used by the directory stream routines described in *directory(3C)*.

The following data types are defined:

DIR A structure containing information about an open directory stream.

struct dirent A structure defining the format of entries returned by the *readdir* function (see *directory(3C)*).

The **struct dirent** structure includes the following members:

```
char d_name[MAXNAMLEN+1]; /* name of directory entry */
ino_t d_ino;               /* file serial number */
short d_namlen;           /* length of string in d_name */
short d_reclen;           /* length of this record */
```

The constant **MAXNAMLEN** is defined in **<dirent.h>**.

Note that the **d_reclen** entry is used internally to represent the offset from the current entry to the next valid entry. Therefore, **d_reclen** is not the length of the current entry, but the length of the current record where a record is an entry plus any currently unused space between the current entry and the next valid entry. The unused space between valid **dirent** entries results from changes in a directory's contents, such as the deletion of files and other directories.

This file also contains external declarations for the functions in the *directory(3C)* package.

AUTHOR

dirent.h was developed by AT&T and HP.

SEE ALSO

directory(3C), *ndir(5)*.

STANDARDS CONFORMANCE

<dirent.h>: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

dld.sl - dynamic loader

MULTITHREAD USAGE

The dynamic loader is thread-safe.

DESCRIPTION

The `/usr/lib/pa20_64/dld.sl` program is the PA64 dynamic loader. The `/usr/lib/dld.sl` program is the PA32 dynamic loader. In programs that use shared libraries, `dld.sl` is invoked automatically at startup time by `exec` on PA64 and by the startup file `crt0.o` on PA32. Identical copies of `crt0.o` are kept in both `/opt/langtools/lib` and `/usr/ccs/lib` directories. The dynamic loader is, itself, a shared library, although it defines no symbols for use by user programs.

Shared Libraries

Shared libraries are executable files created with the `-b` option to `ld` (see `ld(1)`). They must contain position-independent code (PIC) that can be mapped anywhere in the address space of a process and executed with minimal relocation. PIC can use PC-relative addressing modes and/or linkage tables. It is generated by default by the compilers on PA64 and by specifying the `+z/+Z` options on PA32. See the `+help` option to `ld(1)` or the *HP-UX Linker and Libraries User's Guide* manual for details on writing PIC in assembly language.

Incomplete Executables

An executable program linked with one or more shared libraries is called an **incomplete executable**.

When creating an executable (`a.out`) file from object files and libraries, the linker does not copy text (code) or data from the shared library into the output file. Instead, the dynamic loader maps the library into the address space of the process at run time. The linker binds all program references to shared library routines and data to entries in a linkage table, and relies on the dynamic loader to fill in the linkage table entries once the libraries have been mapped. This linkage table serves as a jump table for function calls.

Thread local storage is now supported. The dynamic loader will tally each shared library's thread local storage size as well as the program's thread local storage size. When all libraries are loaded, on PA32 the dynamic loader will either call a thread routine to set the thread pointer and allocate space for the initial thread or will call `mmap()` and `lwp_setprivate()` to allocate the space and setup the thread pointer. On PA64, the dynamic loader will invoke an initializer in the system library `libc` which will do the thread initialization, allocation of the initial thread, and set the thread pointer.

On previous PA32 releases, shared library data items referenced by the program were copied into the program executable file so that the data references could be resolved statically. Beginning with the Series 700/800 10.0 release, references to shared library data from the `a.out` are included in a linkage table and are resolved at run time.

Loading

An incomplete executable contains a list of path names of the shared libraries searched at link time. At run time, the dynamic loader attaches to the process all shared libraries that were linked with the program. The dynamic loader will attempt to load each library from the same directory in which it was found at link time. It is possible to change the shared library run time search path by specifying a dynamic path list. See *PA32 Dynamic Path List* and/or *PA64 Dynamic Path List*.

The text segment of a library is shared among all processes that use it. The data and bss segments are shared on a page-by-page basis. When a process first accesses (reads or writes) a data or bss page, a copy of that page is made for the process.

PA32 Dynamic Path List

There are two ways to specify a dynamic path list :

- by storing a directory path list in the executable using the `+b path_list` option to `ld`,
- by linking the executable with `ld` option `+s`, enabling the executable to use the path list defined by the `SHLIB_PATH` environment variable at run time.

The path list is a list of one or more path names separated by colons (:). The dynamic path list will work only for libraries specified with the `-l` or `-l:` options to `ld`. However, it can be enabled for libraries specified with a full path name using the `-l` option to `chatr` (see `chatr(1)`). If both `+s` and `+b` are used, their relative order on the command line indicates which path list will be searched first in compatibility mode. See the `+help` option to `ld(1)` or the *HP-UX Linker and Libraries User's Guide* manual for more

details.

PA64 Dynamic Path List

For standard mode libraries (libraries built or linked with **ld +std**), the dynamic loader will use dynamic path searching to find shared libraries whose names appear in a library list of the program or loaded shared libraries with no embedded **/** character. Dynamic path searching is enabled by default for these standard mode libraries or executables. If **ld +noenvvar** is specified, the dynamic loader will not look at any dynamic path environment variables to find dependent shared libraries. This limits the dynamic path searching to the value of **rpath** and the default directories **/usr/lib/pa20_64** and **/usr/ccs/lib/pa20_64**.

For compatibility mode libraries (libraries built or linked with **ld +compat**), the dynamic loader will only do dynamic path searching for these libraries if they were linked with **-l** or **-l:** and one of these were specified:

- **ld +s**
- **ld +b**
- **chatr +s enable**

There are several ways to specify a dynamic path list :

- By storing a directory path list in the executable using the **+b path_list** option to **ld**.
- By *not* specifying **ld +b** and letting the linker set the **rpath** value to a concatenation of the **ld -L path_list** followed by the value of the environment variable **LPATH** followed by the default directories **/usr/lib/pa20_64** and **/usr/ccs/lib/pa20_64**. This is for standard mode shared libraries only.
- By storing a directory **path_list** in the environment variables **LD_LIBRARY_PATH** and/or **SHLIB_PATH**. For compatibility mode shared libraries and executables, the directory **path_list** should only be put in the **SHLIB_PATH** environment variable.

The path list is a list of one or more path names separated by colons (:). The dynamic path list will work only for libraries specified with the **-l** or **-l:** options to **ld**. However, it can be enabled for libraries specified with a full path name using the **-l** option to **chatr** (see *chatr(1)*). If both **+s** and **+b** are used, their relative order on the command line indicates which path list will be searched first in compatibility mode. See the **+help** option to *ld(1)* or the *HP-UX Linker and Libraries User's Guide* manual for more details.

The dynamic loader will use these rules when determining which dynamic path list to use:

- If **ld +noenvvar** was specified and **ld +b** and **ld +compat** were *not* specified, then the only dynamic path searching that can be done is to look at the **path_list** in **rpath** followed by the default directories **/usr/lib/pa20_64** and **/usr/ccs/lib/pa20_64**.
- If **ld +compat** was specified and **ld +b** and **ld +s** were *not* specified, no shared library is subject to dynamic path searching.
- If the **ld +compat** and **ld +b** options are *not* specified, then the **path_list** in the **LD_LIBRARY_PATH** environment variable is searched, followed by the **path_list** in the **SHLIB_PATH** environment variable, followed by the **path_list** in **rpath**, followed by the default directories **/usr/lib/pa20_64** and **/usr/ccs/lib/pa20_64**.
- If the **ld +compat**, **ld +b**, and **ld +s** are specified, then use the relative ordering of **ld +b** and **ld +s** to determine if the dynamic loader should use the **path_list** in **rpath** before **SHLIB_PATH** followed by library as specified in the library list. If **ld +b** is specified first, use the **path_list** in **rpath** first.

The rules change slightly when looking for dependent shared libraries.

- For standard mode libraries, the **path_list** in the **LD_LIBRARY_PATH** environment variable is searched first, followed by the **path_list** in the **SHLIB_PATH** environment variable, followed by the value in the parent shared library's **rpath**, followed by the default directories **/usr/lib/pa20_64** and **/usr/ccs/lib/pa20_64**. The ancestors of a parent shared library may contain a **path_list** in **rpath**, but this is ignored when searching for dependent shared libraries of this parent. Only the parent's **rpath** is used.
- For compatibility mode libraries, the search is the same as for parent shared libraries, except **rpath** can be passed from parent shared libraries to child dependent shared libraries to that child's

dependents, et cetera.

Binding

The dynamic loader also resolves symbolic references between the executable and libraries. By default, function calls are trapped via the linkage table and bound on first reference. References to data symbols and other absolute address references cannot be trapped. They are bound on the first resolution of a function call that could potentially reference the object.

If the **-B immediate** option to **ld** is used, the loader binds all necessary references at startup time. This increases the startup cost of a program, but ensures that no more binding operations will be required later. Thus, better real-time response may result, and the risk of a later abort due to unresolved externals is eliminated.

The **fastbind** tool can be used to improve the start-up time of programs that use shared libraries (incomplete executables). The **fastbind** tool performs analysis on the shared library routines and data used to bind the symbols and stores this information in the executable file. The dynamic loader will notice that this information is available, and it will use this **fastbind** information to bind the symbols instead of the standard search method. For more details refer to *fastbind(1)* and the **+help** option to *ld(1)* or the *HP-UX Linker and Libraries User's Guide* manual.

Breadth-first Searching

This is only available on PA64. By default, the dynamic loader will do breadth-first searching when binding symbols. If the incomplete executable was linked with **+compat** or if a **shl_load()** is being executed, then depth-first searching is used. Breadth-first searching specifies that the dynamic loader will look for symbols starting with the incomplete executable followed by all loaded shared libraries in a left to right order until the symbol is found. For example, the incomplete executable is searched followed by all libraries in its library load list. Then the dependent shared libraries of the first library in the library load list is searched, followed by the dependent shared libraries of the second library in the list, et cetera.

Depth-first Searching

This is the only search behavior on PA32 and is used on PA64 if doing a **shl_load()** or if the incomplete executable was linked with **+compat**. The dynamic loader will search the incomplete executable followed by the first library in its library load list. The first dependent library of this library is then searched, followed by the first dependent of this dependent, and so on. When there are no more dependents, the siblings and their dependents are searched until eventually the second library in the program's library load list is searched, followed by the first dependent of this library, et cetera.

Version Control

Since code from a shared library is mapped at run time from a separate shared library file, modifications to a shared library may alter the behavior of existing executables. In some cases, this may cause programs to operate incorrectly. Two means of version control is provided to solve this problem.

Intra-Library Versioning

This is available on PA32 only. Whenever an incompatible change is made to a library interface, both versions of the affected module or modules are included in the library. A mark indicating the date (month/year) the change was made is recorded in the new module via the pragma **HP_SHLIB_VERSION** in C, or the compiler directive **SHLIB_VERSION** in Fortran and Pascal. This date applies to all symbols defined within the module. A high water mark giving the date of the latest incompatible change is recorded in the shared library, and the high water mark for each library linked with the program is recorded in the incomplete executable file. At run time, the dynamic loader checks the high water mark of each library and loads the library only if it is at least as new as the high water mark recorded at link time. When binding symbolic references, the loader chooses the latest version of a symbol that is not later than the high water mark recorded at link time. These two checks help ensure that the version of each library interface used at run time is the same as was expected at link time. Intra-library versioning may be removed in a future release.

Library-level Versioning

The second way for users to version their libraries is by using a new naming convention, **libname.n** where **n** is a numeral that is incremented with every new release of the library. When using the new naming scheme, users must specify an internal name for the shared library by using the **+h internal_name** option to **ld** when building the shared library. This internal name is recorded in each incomplete executable or shared library that links with the shared library.

At run time, the loader will look at the library list recorded in the incomplete executable file or shared library. For each library in the list that was not an internal name, the dynamic loader will look for a `.0` version of the library (e.g. `libname.0`) to load. If it does not find this version, it will look for the library name that is recorded in the list.

PA32 Explicit Loading and Binding

The duties of the dynamic loader as described above are all performed automatically, although they can be controlled somewhat by appropriate options to `ld`. The dynamic loader can also be accessed programmatically. The reserved symbol `__dld_loc`, which is defined in `crt0.o`, points to a jump table within the dynamic loader. The routines described under `shl_load(3X)` provide a portable interface that allows the programmer to explicitly attach a shared library to the process at run time, to calculate the addresses of symbols defined within shared libraries, and to detach the library when done.

PA64 Explicit Loading and Binding

The duties of the dynamic loader as described above are all performed automatically, although they can be controlled somewhat by appropriate options to `ld`. The dynamic loader can also be accessed programmatically. The routines described under `shl_load(3X)`, `dlclose(3C)`, `dlderror(3C)`, `dlget(3C)`, `dlmodinfo(3C)`, `dlopen(3C)`, and `dlsym(3C)` provide a portable interface that allows the programmer to explicitly attach a shared library to the process at run time, to calculate the addresses of symbols defined within shared libraries, and to detach the library when done.

DIAGNOSTICS

If the dynamic loader is not present, or cannot be invoked by the process for any reason, an error message is printed to standard error and the process terminates with a non-zero exit code.

These errors fall into two basic categories: errors in attaching a shared library, and errors in binding symbols. The former can occur only at process startup time but the latter can occur at any time during process execution unless the `-B immediate` option is used with `ld`. Possible errors that can occur while attaching a shared library include library not present, library not executable, library corrupt, high water mark too low, or insufficient room in the address space for the library. Possible errors that can occur while binding symbols include symbol not found (unresolved external), or library corrupt.

When using the explicit load facilities of the dynamic loader, these types of errors are not considered fatal. Consult `shl_load(3X)`, `dlclose(3C)`, `dlget(3C)`, `dlgetname(3C)`, `dlmodinfo(3C)`, `dlopen(3C)`, and `dlsym(3C)` for more information. On PA64, to see error messages, use the `dlderror()` routine. This routine will print the last error message recorded by the dynamic loader.

WARNINGS

The startup cost of the dynamic loader is significant, even with deferred binding, and can cause severe performance degradation in processes dominated by startup costs (such as simple “hello world” programs). In addition, position-independent code is usually slower than normal code, so performance of a program may be adversely affected by the presence of PIC in shared libraries. However, the advantages of decreased disk space usage and decreased memory requirements for executables should outweigh these concerns in most cases.

There are rare cases where the behavior of a program differs when using shared libraries as opposed to archive libraries. This happens primarily when relying on undocumented and unsupported features of the compilers, assembler, and linker. See the `+help` option to `ld(1)` or the *HP-UX Linker and Libraries User's Guide* manual for more details.

The library developer is entirely responsible for version control and must be thorough in identifying incompatible changes to library interfaces. Otherwise, programs may malfunction unexpectedly with later versions of the library. There is little an application user can do if version control is not handled properly by the library developer. The application developer can usually resolve problems by modifying the source code to use the new interfaces then recompiling and relinking against the new libraries.

By default, most warnings are not reported by the dynamic loader.

On PA32, if you wish to see all of the messages, set the environment variable `_HP_DLDOPTS` to contain one or more options. The following options are supported:

- warnings** Display additional dynamic loader warning messages. Some of these include:
 - Symbols of the same name but different types, such as CODE and DATA. See the *WARNINGS* section in `ld(1)` for more details on this warning.

- Using certain flags or routines described in *shl_load(3X)*.

-fbverbose See *fastbind(1)*.

-nofastbind See *fastbind(1)*.

On PA64, if you wish to see all error messages, set the environment variable **DLD_VERBOSE_ERR** to true.

AUTHOR

The `/usr/lib/dld.sl` and `/usr/lib/pa20_64/dld.sl` shared libraries were developed by HP.

SEE ALSO

System Tools:

<i>as(1)</i>	translate assembly code to machine code
<i>CC(1)</i>	invoke the HP-UX C++ compiler
<i>cc(1)</i>	invoke the HP-UX C compiler
<i>chatr(1)</i>	change program's internal attributes
<i>f77(1)</i>	invoke the HP-UX FORTRAN compiler
<i>fastbind(1)</i>	invoke the fastbind tool
<i>ld(1)</i>	invoke the link editor
<i>pc(1)</i>	invoke the HP-UX Pascal compiler

Miscellaneous:

<i>a.out(4)</i>	assembler, compiler, and linker output
<i>dlclose(3C)</i>	unload a shared library previously loaded by <i>dlopen()</i>
<i>dlerror(3C)</i>	print the last error message recorded by <i>dld</i>
<i>dlget(3C)</i>	return information about a loaded module
<i>dlgetname(3C)</i>	return the name of the storage containing a load module
<i>dlmodinfo(3C)</i>	return information about a loaded module
<i>dlopen(3C)</i>	load a shared library
<i>dlsym(3C)</i>	get the address of a symbol in a shared library
<i>shl_load(3X)</i>	load/unload shared libraries

Texts and Tutorials

HP-UX Linker and Libraries Online User Guide
(See the **+help** option to *ld(1)*)

HP-UX Linker and Libraries User's Guide
(See *manuals(5)* for ordering information)

NAME

environ - user environment

DESCRIPTION

An array of strings called the **environment** is made available by *exec(2)* when a process begins. By convention, these strings have the form *name=value*. The following names are used by various commands (listed in alphabetical order):

HOME Name of the user's login directory, set by *login(1)* from the password file (see *passwd(4)*).

LANG Identifies the user's requirements for native language, local customs and coded character set, if the environment variables `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME` are unset or null.

The format of `LANG` is:

```
LANG=language[_territory][.codeset]
```

The valid values for `LANG` are supported locales. (See *lang(5)*.) Native Language Support (NLS) is initiated at run-time by calling *setlocale(3C)*. The following call to *setlocale* binds the execution of a program to the user's language requirements:

```
setlocale(LC_ALL, "");
```

This *setlocale* call initializes the program locale from the environment variables associated with *setlocale*. `LANG` provides the necessary defaults if any of the category-specific environment variables are not set or set to the empty string.

The `LANG` environment variable can have a maximum length of `SL_NAME_SIZE` bytes (see header file **<locale.h>**).

LANGOPTS Defines language options for mode and data order in the form:

```
LANGOPTS=[mode][_order]
```

`LANGOPTS` values are given in English as an ASCII character string. *mode* describes the mode of a file where **l** (ell) represents Latin mode and **n** represents non-Latin mode. Non-Latin mode is assumed for values other than **l** and **n**. *order* describes the data order of a file where **k** is keyboard order and **s** is screen order.

LC_ALL Determines the values for all locale categories. The value of `LC_ALL` has precedence over any of the other environment variables `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`, and `LANG`.

LC_COLLATE, **LC_CTYPE**, **LC_MESSAGES**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME**

determines the user's requirements for language, territory, and codeset with respect to character collation, character classification and conversion, output messages, currency symbol and monetary value format, numeric data presentation, and time formats, respectively. If `LC_ALL` and any of these are not defined in the environment, `LANG` provides the defaults.

Syntax for the environment variables `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME` is:

```
language[_territory][.codeset][@modifier]
```

The *language* field conforms with ISO 639 standard for language names and the *territory* field conforms with the ISO 3166 territory names. For a list of the locale names, see *lang(5)*.

The *@modifier* field allows the user to select between more than one value of a category within the same language definition. HP-UX does not currently provide locales with modifiers.

The values of the locale categories are determined by a precedence order; the first condition met below determines the value:

1. If the `LC_ALL` environment variable is defined and is not null, the value of `LC_ALL` is used.
2. If the `LC_*` environment variable (`LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`) is defined and is not null, the value of the environment variable is used to initialize the category that corresponds to the environment variable.
3. If the `LANG` environment variable is defined and is not null, the value of the `LANG` environment variable is used.

4. If the LANG environment variable is not set or is set to the empty string, the POSIX/C default locale is used. (See *lang*(5).)

LC_COLLATE

Determines the locale category for character collation. It determines collation information for regular expressions and sorting, including equivalence classes and multi-character collating elements, in various utilities and *strcoll*(3C) and *strxfrm*(3C).

LC_CTYPE

Determines the locale category for character classification (such as alphabetic, digit, upper-case. See *isalpha*(3C), *isdigit*(3C), and *isupper*(3C)), character conversion. (see *toupper*(3C), *tolower*(3C)), and the interpretation of text as single-byte or multi-byte characters,

LC_MESSAGES

Determines the locale category for processing affirmative and negative responses and the language and cultural conventions in which diagnostic and informative messages should be written. It may also affect the behavior of *catopen*(3C) in determining the message catalog to open.

LC_MONETARY

Determines the locale category for monetary-related numeric formatting information.

LC_NUMERIC

Determines the locale category for numeric formatting information (such as the thousands separator and the radix character) in various utilities as well as the formatted I/O operations in *printf*(3S) and *scanf*(3S) and the string conversion functions in *strtod*(3C).

LC_TIME

Determines the locale category for date and time formatting information. It affects the behavior of time functions in *strtime*(3C).

MANPATH

Contains a colon-separated list of directory prefixes to be searched by *man*(1) for manual entries. Upon logging in, */etc/profile* (or */etc/csh.login*) sets **MANPATH=/usr/share/man:/usr/contrib/man:usr/local/man**.

MANPATH uses the same syntax as the PATH environment variable, with the addition of recognizing the specifiers %L, %l, %t, and %c as used in the NLSPATH environment variable. See NLSPATH below for a description of these specifiers. This provides a way to specify paths to locale-specific manual entries.

It is assumed that each of the prefixes given in MANPATH contain subdirectories of the form **man***, **man*.Z**, **cat*** and **cat*.Z**. (See *man*(1), *catman*(1M), and *fixman*(1M).)

NLSPATH

Contains a sequence of pseudo-pathnames used by *catopen*(3C) when attempting to locate message catalogs. Each pseudo-pathname contains a name template consisting of an optional path prefix, one or more substitution field descriptors, a file name and an optional file name suffix. For example:

```
NLSPATH="/system/nlslib/%N.cat"
```

defines that *catopen*(3C) should look for all message catalogs in the directory **/system/nlslib**, where the catalog name should be constructed from the *name* parameter passed to *catopen*(3C) (%N) with the suffix **.cat**.

Field descriptors consist of a % followed by a single character. Field descriptors and their substitution values are:

%N	The value of the <i>name</i> parameter passed to <i>catopen</i> (3C).
%L	The value of LC_MESSAGES.
%l	The <i>language</i> element from LC_MESSAGES.
%t	The <i>territory</i> element from LC_MESSAGES.
%c	The <i>codeset</i> element from LC_MESSAGES.
%%	Replaced by a single %.

For example, given:

```
NLSPATH="/system/nlslib/%L/%N.cat"
```

catopen(3C) attempts to open the file **/system/nlslib/\$LC_MESSAGES/name.cat** as a message catalog.

A null string is substituted if the specified value is not defined. Separators are not included in %t and %c substitutions. Note that a default value is not supplied for %L. If LC_MESSAGES is

not set and NLSPATH had the value in the previous example, *catopen*(3C) would attempt to open the file `/system/nlslib/name.cat` as a message catalog.

Path names defined in NLSPATH are separated by colons (:). A leading colon or two adjacent colons (::) is equivalent to specifying %N. For example, given:

```
NLSPATH=":%N.cat:/nlslib/%L/%N.cat"
```

catopen(3C) with the *offlag* parameter set to `NL_CAT_LOCALE` will attempt to open the following files in the indicated order: *.name*, *.name.cat*, and `/nlslib/SLC_MESSAGES/name.cat`. The first file successfully opened is taken as the message catalog.

A default pseudo-pathname defined by the system is effectively appended to NLSPATH and used by *catopen*(3C) whenever a message catalog cannot be opened in any of the user defined pseudo-paths. This system-wide default path is:

```
/usr/lib/nls/msg/%L/%N.cat:/usr/lib/nls/%l/%t/%c/%N.cat
```

PAGER PAGER indicates the paginator through which output from certain commands is piped. Its value must be a string specifying the complete command line of the desired paginator. Two examples are:

```
PAGER="more -cs"
PAGER="pg -c"
```

PAGER affects several commands, including *man*(1) and the interactive mailers. Some of the affected commands provide alternate means of selecting a pager in case there is a conflict. See the individual manual entries for details.

PATH PATH indicates the sequence of directory prefixes that *sh*(1), *time*(1), *nice*(1), *nohup*(1), and others search when looking for a file known by an incomplete path name. Prefixes are separated by colons (:). *Login*(1) sets `PATH=/usr/bin`.

TERM TERM identifies the kind of terminal for which output is to be prepared. This information is used by commands such as *vi*(1) and *mm*(1), which can exploit special capabilities of that terminal.

TZ TZ sets time zone information. TZ can be set using the format:

```
[:]STDoffset[DST[offset][,rule]]
```

where:

STD and DST Three or more bytes that designate the standard time zone (STD) and summer (or daylight-savings) time zone (DST). STD is required. If DST is not specified, summer time does not apply in this locale. Any characters other than digits, comma (,), minus (-), plus (+), or ASCII NUL are allowed.

offset offset is the value that must be added to local time to arrive at Coordinated Universal Time (UTC). Offset is of the form :

```
hh[:mm[:ss]]
```

Hour (hh) is any value from 0 through 23. The optional minutes (mm) and seconds (ss) fields are a value from 0 through 59. The hour field is required. If offset is preceded by a -, the time zone is east of the Prime Meridian. A + preceding offset indicates that the time zone is west of the Prime Meridian. The default case is west of the Prime Meridian.

rule rule indicates when to change to and from summer (daylight-savings) time. The rule has the form :

```
date/time,date/time
```

where the first *date/time* specifies when to change from standard to summer time, and the second *date/time* specifies when to change back. The *time* field is expressed in current local time.

The form of *date* should be one of the following :

Jn Julian day *n* (1 through 365). Leap days are not counted. February 29 cannot be referenced.

n The zero-based Julian day (0 through 365). Leap days are counted. February 29 can be referenced.

Mm.n.d The *d* day (0 through 6) of week *n* (1 through 5) of month *m* (1 through 12) of the year. Week 5 refers to the last day *d* of month *m*. Week 1 is the week in which the first day of the month falls. Day 0 is Sunday.

time *Time* has the same format as *offset* except that no leading sign ("- or "+) is allowed. The default, if *time* is not given, is 02:00:00.

While the STD field and the offset field for STD must be specified, if the DST field is also provided, the system will supply default values for other fields not specified. These default values come from file `/usr/lib/tztab` (see *tztab(4)*), and, in general, reflect the various historical dates for start and end of summer time.

Additional names may be placed in the environment by the *export* command and "name=value" arguments in *sh(1)*, or by *exec(2)*. It is unwise to add names that conflict with the following shell variables frequently exported by `.profile` files: `MAIL`, `PS1`, `PS2` and `IFS`.

The environment of a process is accessible from C by using the global variable:

```
char **environ;
```

which points to an array of pointers to the strings that comprise the environment. The array is terminated by a null pointer.

WARNINGS

Some HP-UX commands and library routines do not use the `LANG`, `LC_COLLATE`, `LC_CTYPE`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`, or `LANGOPTS` environment variables. Some commands do not use message catalogs, so `NLSPATH` does not affect their behavior. See the EXTERNAL INFLUENCES section of specific commands and library routines for implementation details.

NOTES

Coordinated Universal Time (UTC) is equivalent to Greenwich Mean Time (GMT).

AUTHOR

Environ was developed by AT&T and HP.

SEE ALSO

env(1), *login(1)*, *sh(1)*, *exec(2)*, *catopen(3C)*, *ctime(3C)*, *getenv(3C)*, *setlocale(3C)*, *profile(4)*, *lang(5)*, *term(5)*, *tztab(4)*.

STANDARDS CONFORMANCE

environ: AES, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

fcntl - file control options

SYNOPSIS

```
#include <sys/types.h>
#include <fcntl.h>
```

DESCRIPTION

The **fcntl()** function provides for control over open files. The **<fcntl.h>** include file describes *requests* and *arguments* to **fcntl()** and **open()**. See *fcntl(2)* and *open(2)*.

The access modes set by **open()** and accessed by **fcntl()** are:

```
O_RDONLY
O_WRONLY
O_RDWR
```

The mask for file access modes is:

```
O_ACCMODE
```

The file status flags set by **open()** or **fcntl()** and accessed by **fcntl()** are:

O_NDELAY	Nonblocking I/O.
O_NONBLOCK	POSIX-style nonblocking I/O.
O_APPEND	Append (writes guaranteed at the end).
O_DSYNC	Write through cache for data.
O_SYNC	Write through cache for data and attributes.
O_RSYNC O_DSYNC	Write through cache for data on reads and writes.
O_RSYNC O_SYNC	Write through cache for data and attributes on reads and writes.
O_LARGEFILE	When the filesystem is mounted as large files enabled, the O_LARGEFILE option allows the file to grow over 2 GB.

The flag **O_SYNCIO** is a synonym for **O_SYNC** and is defined for backward compatibility in **<fcntl.h>**.

The flag values accessible only to **open()** are:

O_CREAT	Open with file create (uses third open arg).
O_TRUNC	Open with truncation.
O_EXCL	Exclusive open.
O_NOCTTY	Do not assign a controlling terminal.

The requests for **fcntl()** are:

F_DUPFD	Duplicate file descriptor.
F_GETFD	Get file descriptor flags.
F_SETFD	Set file descriptor flags.
F_GETFL	Get file flags.
F_SETFL	Set file flags.
F_GETLK	Get blocking file lock.
F_SETLK	Set or clear file locks and fail on busy.
F_SETLKW	Set or clear file locks and wait on busy.

The file descriptor flag for **F_GETFD**, **F_SETFD** is:

```
FD_CLOEXEC
```

The file segment locking control structure **struct flock**, includes the following members:

```
short l_type;      /* F_RDLCK, F_WRLCK or F_UNLCK */
short l_whence;    /* Flag - see lseek(2) */
off_t l_start;     /* Relative offset in bytes */
off_t l_len;       /* Size; if 0 then until EOF */
pid_t l_pid;       /* By F_GETLK - process holding lock */
```

The file segment locking types are:

F_RDLCK	Read lock.
F_WRLCK	Write lock.
F_UNLCK	Remove locks.

SEE ALSO

fcntl(2), open(2).

STANDARDS CONFORMANCE

<fcntl.h>: AES, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1


f

NAME

fenv - floating-point environment macros and functions

SYNOPSIS

```
#include <fenv.h>
```

DESCRIPTION

The header `<fenv.h>` declares two types and several macros and functions to provide access to the floating-point environment. The *floating-point environment* refers collectively to the floating-point status flags and control modes. A *floating-point status flag* is a system variable whose value is set as a side effect of the arithmetic to provide auxiliary information. A *floating-point control mode* is a system variable whose value may be set by the user to affect the subsequent behavior of the arithmetic; on HP 9000 systems the control modes include the rounding direction mode, the underflow mode, and the trap enables.

The following types are defined:

fenv_t	Represents the entire floating-point environment.
fexcept_t	Represents the floating-point exception flags collectively.

The following macros represent the floating-point status flags. They are defined as integral constant expressions.

FE_INEXACT	The inexact exception.
FE_DIVBYZERO	The division-by-zero exception.
FE_UNDERFLOW	The underflow exception.
FE_OVERFLOW	The overflow exception.
FE_INVALID	The invalid operation exception.
FE_ALL_EXCEPT	The bitwise OR of all exception macros.

The following macros represent the rounding direction modes. They are defined as integral constant expressions.

FE_TONEAREST	The round-to-nearest rounding direction mode.
FE_UPWARD	The round-toward-positive-infinity rounding direction mode.
FE_DOWNWARD	The round-toward-negative-infinity rounding direction mode.
FE_TOWARDZERO	The round-toward-zero rounding direction mode.

The following macro is defined as a pointer to const-qualified **fenv_t**:

FE_DFL_ENV	The default floating-point environment.
-------------------	---

The `<fenv.h>` header file and its contents have been approved for inclusion in the C9X standard. The HP implementation adds four HP-specific functions to the approved contents: **fegetflushtozero()**, **fesetflushtozero()**, **fegettrapenable()**, and **fesettrapenable()**.

FILES

`/usr/include/fenv.h`

SEE ALSO

feclearexcept(3M), **fegetexceptflag(3M)**, **feraiseexcept(3M)**, **fesetexceptflag(3M)**, **fetestexcept(3M)**, **fegetround(3M)**, **fesetround(3M)**, **fegetenv(3M)**, **feholdexcept(3M)**, **fesetenv(3M)**, **feupdateenv(3M)**, **fegetflushtozero(3M)**, **fesetflushtozero(3M)**, **fegettrapenable(3M)**, **fesettrapenable(3M)**.

NAME

fs_wrapper - configuration and binary files used by file system administration commands

SYNOPSIS

```
ff [-F FStype] ...
fsck [-F FStype] ...
fsdb [-F FStype] ...
labelit [-F FStype] ...
mkfs [-F FStype] ...
mount [-F FStype] ...
ncheck [-F FStype] ...
newfs [-F FStype] ...
quot [-F FStype] ...
quotacheck [-F FStype] ...
volcopy [-F FStype] ...
```

DESCRIPTION

The commands listed in the SYNOPSIS can operate on different types of file systems. Each command (except for **mount**) reads file system specific configuration files that control the command's behavior, and invokes a file system specific binary file to do the actual work. *FStype* is the file system type as optionally specified on the command line. If *FStype* is not given, then the file system type is determined from **/etc/fstab** by matching an entry in this file with a device **special** provided with the command (see individual commands for details of usage).

Administrators may also define a default file system type for the above commands via the file **/etc/default/fs**. If this file exists, and contains the line:

```
LOCAL=FStype
```

(e.g., **LOCAL=hfs**), then the above commands will assume the *FStype* given in **/etc/default/fs**, unless an *FStype* is provided on the command line or is in **/etc/fstab**. The default file system specification is provided to maintain compatibility with pre-10.0 invocations of the commands.

See the FILES section for a list of the files used.

WARNINGS

The configuration files **/sbin/lib/mfsconfig.d/*FStype*** are supplied by HP or by other file system vendors. They are not meant to be edited by System Administrators. Corruption or removal of these files may lead to strange behavior, including the inability to boot.

The format of the configuration file is subject to change.

The file system specific binary files are not normally executed directly. However, if the configuration files become unusable, direct execution of these binary files may be a useful step in repairing and running the system again. The binary files accept the same arguments as the commands by which they are executed.

The **mount** command is a special case. This command currently does not read a configuration file, and does not execute a file system specific binary file if *FStype* is **cdfs**, **hfs**, **nfs**, or **lofs**. The binary that handles these *FStypes* also processes other *FStypes* and calls the file system specific command if appropriate.

For historical reasons, the **hfs** binary files also handle **nfs** and **cdfs**, so there are no separate binary files for the latter two file systems.

The commands (except **mount**) will not work if they are renamed, because they are symbolically linked to a single executable (**/sbin/fs_wrapper**).

FILES

FStype is the file system type as optionally specified on the command line. *command* is the name of the command.

<i>/sbin/fs / FStype / command</i>	File system specific binary files for the fsck , fsdb , mkfs , mount , and newfs commands. There may be additional file system specific binary files in this directory that are not associated with fs_wrapper .
<i>/usr/sbin/fs / FStype / command</i>	File system specific binary files for the remaining commands. There may also be other file system specific binary files in this directory that are not associated with fs_wrapper .
<i>/sbin/lib/mfsconfig.d / FStype</i>	Configuration files for each file system type.
<i>/etc/default/fs</i>	File in which the default file system type can be defined. If this file does not exist, there is no default file system type.
<i>/etc/fstab</i>	Static information about the file systems

SEE ALSO

ff(1M), fsck(1M), fsdb(1M), mkfs(1M), mount(1M), ncheck(1M), newfs(1M), quot(1M), quotacheck(1M), volcopy(1M), fstab(4).

f

NAME

hier - file system hierarchy

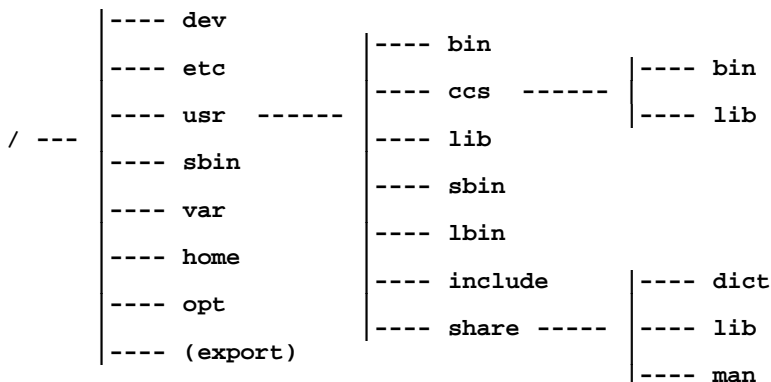
DESCRIPTION

The HP-UX file system is a hierarchical tree organized for administrative convenience. Within the file-system tree structure, distinct areas are provided for files that are private to one machine, files that can be shared by machines, and home directories.

There are two types of files that are shared: those that can be shared by multiple machines of a common architecture, and those that can be shared by all machines. This organization allows sharable files to be stored on one machine (the server), but accessed by many machines (clients).

The following diagram illustrates the file system layout. Note that there are many directories that are not in this diagram, but are discussed below.

Diagram of Directory Layout



The following listing discusses a representative HP-UX directory hierarchy. Some HP-UX applications may add additional directories, which are not shown.

/	Root directory.
/dev	Special files (block and character device files); see <i>mknod</i> (1M).
/etc	Host-specific configuration and administration databases.
/etc/opt	Directory for application-specific configuration files. (Configuration information for optional packages.)
/etc/rc.config.d	Startup configuration files.
/export	Default root of exported file systems. Server only.
/home	Default root for user directories.
/lost+found	Storage directory for connecting detached files; for use by <i>fsck</i> (1M).
/mnt	Mounting point for local file systems.
/net	Mounting point for remote file systems.
/opt	Root of subtree for optional application packages.
/sbin	Essential system commands. Essential commands are defined as executables that are needed to boot the system and mount the file systems. A full complement of utilities is available only after /usr is mounted.
/sbin/init.d	Startup and shutdown scripts.
/sbin/rc0.d	Link files to scripts in /sbin/init.d for entering or leaving run level 0.
/sbin/rc1.d	Link files to scripts in /sbin/init.d for entering or leaving run level 1.

<code>/sbin/rc2.d</code>	Link files to scripts in <code>/sbin/init.d</code> for entering or leaving run level 2.
<code>/sbin/rc3.d</code>	Link files to scripts in <code>/sbin/init.d</code> for entering or leaving run level 3.
<code>/stand</code>	Standalone binaries and kernel configuration files.
<code>/tmp</code>	System-generated temporary files; generally cleared during the boot operation.
<code>/usr</code>	Mount point for sharable user and system administration commands, libraries and documentation.
<code>/usr/bin</code>	Primary location for common utilities and user commands.
<code>/usr/ccs</code>	C compilation system. Tools and libraries used to generate C programs.
<code>/usr/ccs/bin</code>	Development binaries; includes <code>cc</code> , <code>make</code> , <code>strings</code> , etc.
<code>/usr/ccs/lib</code>	Development libraries.
<code>/usr/ccs/lbin</code>	Development backends.
<code>/usr/conf</code>	Kernel configuration files.
<code>/usr/contrib</code>	Directory for user-contributed (unsupported, internal) commands, files, etc. Files in this directory come from outside the local site or organization (for example, from users groups or HP service engineers).
<code>/usr/contrib/bin</code>	User-contributed commands.
<code>/usr/contrib/include</code>	User-contributed include files.
<code>/usr/contrib/lib</code>	User-contributed libraries.
<code>/usr/contrib/man</code>	User-contributed man pages.
<code>/usr/include</code>	Included header files, for C and other programs. Some subdirectories are listed below.
<code>/usr/include/machine</code>	Machine-specific C include files.
<code>/usr/include/nfs</code>	C include files for Network File System (NFS).
<code>/usr/include/sys</code>	Kernel related C-language header files.
<code>/usr/lbin</code>	Directory for backend executables to other commands. A backend executable is an executable that is generally not invoked directly by the user.
<code>/usr/lib</code>	Program libraries, object code and architecture-dependent databases.
<code>/usr/lib/nls</code>	Directory for Native Language Support.
<code>/usr/local</code>	Directory for site-local commands, files, etc. Files under this directory come from inside the local site or organization. See <code>/usr/contrib</code> for non-local unsupported commands and files.
<code>/usr/local/bin</code>	Site-local commands.
<code>/usr/local/lib</code>	Site-local libraries.
<code>/usr/local/man</code>	Site-local man pages.
<code>/usr/newconfig</code>	Default operating system configuration data files. This directory is a directory hierarchy mirroring <code>/</code> . New versions of customizable configuration files and databases are shipped here so as not to overwrite current versions. Files in this directory are copied to regular locations for newly installed systems. System administrators may wish to keep them for later reference.
<code>/usr/old</code>	Files and programs that are being phased out or are obsolete.
<code>/usr/sbin</code>	System administration commands.
<code>/usr/share</code>	Architecture-independent sharable files.
<code>/usr/share/dict</code>	Dictionaries for <code>spell</code> and <code>ispell</code> .

<code>/usr/share/lib</code>	Miscellaneous sharable libraries.
<code>/usr/share/man</code>	Online documentation.
<code>/var</code>	Root of subtree for "varying" files. These are files that are created at runtime and can grow to an arbitrary size. Some examples include log, temporary, transient, and spool files.
<code>/var/adm</code>	System administrative files, such as log files and accounting files. Some of the subdirectories are listed below.
<code>/var/adm/crash</code>	For saving kernel crash dumps.
<code>/var/adm/cron</code>	Directory for <i>cron</i> (1M) queuing.
<code>/var/adm/sw</code>	Default location for software distribution depot.
<code>/var/adm/syslog</code>	Log files generated by <i>syslog</i> . See <i>syslog</i> (3C) and <i>syslogd</i> (1M).
<code>/var/mail</code>	Incoming mail.
<code>/var/news</code>	Local-system news articles for <i>news</i> (1).
<code>/var/opt</code>	Root of subtree for varying files associated with optional software packages.
<code>/var/preserve</code>	Place where <i>ex</i> (1) and <i>vi</i> (1) save lost edit sessions until recovered.
<code>/var/run</code>	Files created when daemons are running. For example, the process ID (PID) file for <i>syslogd</i> , <i>syslog.pid</i> , is put here.
<code>/var/spool</code>	Miscellaneous directories for printer spooling, mail delivery, <i>cron</i> (1M), etc.
<code>/var/spool/cron</code>	<i>cron</i> (1M) and <i>at</i> (1) spooling files.
<code>/var/spool/lp</code>	Printer spool files.
<code>/var/spool/mqueue</code>	Outgoing mail and log files containing messages from the mail system.
<code>/var/spool/uucp</code>	UUCP spool directory.
<code>/var/tmp</code>	Application-generated temporary files. This directory generally is not cleared between system reboots.
<code>/var/uucp</code>	UUCP administration files.

DEPENDENCIES

Some directories include commands or files not supported on all HP-UX implementations.

SEE ALSO

find(1), *grep*(1), *ls*(1), *whereis*(1).

NAME

hostname - host name resolution description

DESCRIPTION

Hostnames are domains. A domain is a hierarchical, dot-separated list of subdomains. For example, the machine **monet**, in the **Berkeley** subdomain of the **EDU** subdomain of the Internet Domain Name System would be represented as

monet.Berkeley.EDU

(with no trailing dot).

Hostnames are often used with network client and server programs, which must generally translate the name to an address for use. (This task is usually performed by the library routine *gethostbyname(3)*.)

When NIS or the host table is being used for hostname resolution, the hostname is looked up without modification. When DNS is used, the resolver may append domains to the hostname.

The default method for resolving hostnames by the Internet name resolver is to follow **RFC 1535**'s security recommendations. Actions can be taken by the administrator to override these recommendations and to have the resolver behave the same as earlier, non-RFC 1535 compliant resolvers.

The default method (using RFC 1535 guidelines) follows:

If the name consists of a single component, i.e. contains no dot, and if the environment variable **HOSTALIASES** is set to the name of a file, that file is searched for a string matching the input hostname. The file should consist of lines made up of two strings separated by white-space, the first of which is the host-name alias, and the second of which is the complete hostname to be substituted for that alias. If a case-insensitive match is found between the hostname to be resolved and the first field of a line in the file, the substituted name is looked up with no further processing.

If there is at least one dot in the name, then the name is first tried as is. The number of dots to cause this action is configurable by setting the threshold using the **ndots** option in **/etc/resolv.conf** (default: 1). If the name ends with a dot, the trailing dot is removed, and the remaining name is looked up (regardless of the setting of the 'ndots' option) and no further processing is done.

If the input name does not end with a trailing dot, it is looked up by searching through a list of domains until a match is found. If neither the search option in the **/etc/resolv.conf** file or the **LOCALDOMAIN** environment variable is used, then the search list of domains contains only the full domain specified by the domain option (in **/etc/resolv.conf**) or the domain used in the local hostname (see *resolver(4)*). For example, if the **domain** option is set to **CS.Berkeley.EDU**, then only **CS.Berkeley.EDU** will be in the search list and will be the only domain appended to the partial hostname, *lithium*, making **lithium.CS.Berkeley.EDU** the only name to be tried using the search list.

If the search option is used in **/etc/resolv.conf** or the environment variable, **LOCALDOMAIN**, is set by the user, then the search list will include what is set by these methods. For example, if the **search** option contained

CS.Berkeley.EDU CChem.Berkeley.EDU Berkeley.EDU

then the partial hostname (e.g., *lithium*) will be tried with each domain name appended (in the same order specified). The resulting hostnames that would be tried are:

lithium.CS.Berkeley.EDU
lithium.CChem.Berkeley.EDU
lithium.Berkeley.EDU

The environment variable **LOCALDOMAIN** overrides the **search** and **domain** options, and if both options are present in the resolver configuration file, then only the last one listed is used (see *resolver(5)*).

If the name was not previously tried "as is" (i.e., it fell below the **ndots** threshold or did not contain a dot), then the name, as originally provided, is attempted.

AUTHOR

hostname was developed by the University of California, Berkeley.

SEE ALSO

gethostbyname(3), *resolver(4)*, *named(1M)*, **RFC 1535**.

NAME

inttypes - basic integer data types

SYNOPSIS

```
#include <inttypes.h>
```

DESCRIPTION

This header file defines integer data types of various sizes. By using the data types defined in this header file, developers can be assured that the data types will have the same properties and behavior on different systems.

Since not all implementations are required to support all of the integer sizes defined in this manual page, the proper way to see if a particular size of an integer is supported on the current implementation is to test the symbol that defines its maximum value. For example, if `#ifdef UINT64_MAX` tests false, then that implementation does not support 64-bit unsigned signed integers.

This header file defines the following integer data types for 8, 16, 32, and 64 bits.

```
intmax_t  largest signed integer data type supported by implementation
int8_t    8-bit signed integer
int16_t   16-bit signed integer
int32_t   32-bit signed integer
int64_t   64-bit signed integer
```

```
uintmax_t largest unsigned integer data type supported by implementation
uint8_t    8-bit unsigned integer
uint16_t   16-bit unsigned integer
uint32_t   32-bit unsigned integer
uint64_t   64-bit unsigned integer
```

The following two data types are signed and unsigned integer data types that are large enough to hold a pointer. A pointer can be moved to or from these data types without corruption.

```
intptr_t  signed integer type that is large enough to hold a pointer
uintptr_t unsigned integer type that is large enough to hold a pointer
```

This header file defines the following integer data types for determining the most efficient data types to use for integer values on a particular implementation.

```
intfast_t    most efficient signed integer data type supported by implementation
int_fast8_t  most efficient signed integer of at least 8 bits
int_fast16_t most efficient signed integer of at least 16 bits
int_fast32_t most efficient signed integer of at least 32 bits
int_fast64_t most efficient signed integer of at least 64 bits

uintfast_t   most efficient unsigned integer data type supported by implementation
uint_fast8_t most efficient unsigned integer of at least 8 bits
uint_fast16_t most efficient unsigned integer of at least 16 bits
uint_fast32_t most efficient unsigned integer of at least 32 bits
uint_fast64_t most efficient unsigned integer of at least 64 bits
```

This header file defines the following integer data types for compatibility with systems that do not fit the 16-bit or 32-bit word size model. These data types define the signed and unsigned integers of at least 8, 16, 32, and 64 bits.

```
int_least8_t  smallest signed integer of at least 8 bits
int_least16_t smallest signed integer of at least 16 bits
int_least32_t smallest signed integer of at least 32 bits
int_least64_t smallest signed integer of at least 64 bits

uint_least8_t  smallest unsigned integer of at least 8 bits
uint_least16_t smallest unsigned integer of at least 16 bits
uint_least32_t smallest unsigned integer of at least 32 bits
uint_least64_t smallest unsigned integer of at least 64 bits
```

The following macros define the minimum and maximum values that can be stored in the above data types.

```
INTMAX_MIN  minimum value that can be stored in the largest integer data type
INTMAX_MAX  maximum value that can be stored in the largest signed integer data type
```

UINTMAX_MAX maximum value that can be stored in the largest unsigned integer data type
INTFAST_MIN minimum value that can be stored in the most efficient integer data type
INTFAST_MAX maximum value that can be stored in the most efficient signed integer data type
UINTFAST_MAX maximum value that can be stored in the most efficient unsigned integer data type

INT8_MIN minimum value that can be stored in an `int8_t` data type
INT16_MIN minimum value that can be stored in an `int16_t` data type
INT32_MIN minimum value that can be stored in an `int32_t` data type
INT64_MIN minimum value that can be stored in an `int64_t` data type

INT8_MAX maximum value that can be stored in an `int8_t` data type
INT16_MAX maximum value that can be stored in an `int16_t` data type
INT32_MAX maximum value that can be stored in an `int32_t` data type
INT64_MAX maximum value that can be stored in an `int64_t` data type

UINT8_MAX maximum value that can be stored in an `uint8_t` data type
UINT16_MAX maximum value that can be stored in an `uint16_t` data type
UINT32_MAX maximum value that can be stored in an `uint32_t` data type
UINT64_MAX maximum value that can be stored in an `uint64_t` data type

INT_FAST8_MIN minimum value that can be stored in an `int_fast8_t` data type
INT_FAST16_MIN minimum value that can be stored in an `int_fast16_t` data type
INT_FAST32_MIN minimum value that can be stored in an `int_fast32_t` data type
INT_FAST64_MIN minimum value that can be stored in an `int_fast64_t` data type

INT_FAST8_MAX maximum value that can be stored in an `int_fast8_t` data type
INT_FAST16_MAX maximum value that can be stored in an `int_fast16_t` data type
INT_FAST32_MAX maximum value that can be stored in an `int_fast32_t` data type
INT_FAST64_MAX maximum value that can be stored in an `int_fast64_t` data type

INT_LEAST8_MIN minimum value that can be stored in an `int_least8_t` data type
INT_LEAST16_MIN minimum value that can be stored in an `int_least16_t` data type
INT_LEAST32_MIN minimum value that can be stored in an `int_least32_t` data type
INT_LEAST64_MIN minimum value that can be stored in an `int_least64_t` data type

INT_LEAST8_MAX maximum value that can be stored in an `int_least8_t` data type
INT_LEAST16_MAX maximum value that can be stored in an `int_least16_t` data type
INT_LEAST32_MAX maximum value that can be stored in an `int_least32_t` data type
INT_LEAST64_MAX maximum value that can be stored in an `int_least64_t` data type

The following macros can be used as formatting options with the `printf()` family of functions (see *printf(3S)*). These macros are used to select the correct formatting option for the integer data types defined earlier in this manual page.

PRId8 d print formatting option for `int8_t`
PRId16 d print formatting option for `int16_t`
PRId32 d print formatting option for `int32_t`
PRId64 d print formatting option for `int64_t`
PRIdMAX d print formatting option for `intmax_t`

PRIdFAST8 d print formatting option for `int_fast8_t`
PRIdFAST16 d print formatting option for `int_fast16_t`
PRIdFAST32 d print formatting option for `int_fast32_t`
PRIdFAST64 d print formatting option for `int_fast64_t`
PRIdFAST d print formatting option for `intfast_t`

PRIdLEAST8 d print formatting option for `int_least8_t`
PRIdLEAST16 d print formatting option for `int_least16_t`
PRIdLEAST32 d print formatting option for `int_least32_t`
PRIdLEAST64 d print formatting option for `int_least64_t`

PRi8 i print formatting option for `int8_t`
PRi16 i print formatting option for `int16_t`
PRi32 i print formatting option for `int32_t`
PRi64 i print formatting option for `int64_t`

PRiFAST8 i print formatting option for `int_fast8_t`
PRiFAST16 i print formatting option for `int_fast16_t`

```

PRIiFAST32    i  print formatting option for int_fast32_t
PRIiFAST64    i  print formatting option for int_fast64_t

PRIiLEAST8    i  print formatting option for int_least8_t
PRIiLEAST16   i  print formatting option for int_least16_t
PRIiLEAST32   i  print formatting option for int_least32_t
PRIiLEAST64   i  print formatting option for int_least64_t

PRIu8         u  print formatting option for uint8_t
PRIu16        u  print formatting option for uint16_t
PRIu32        u  print formatting option for uint32_t
PRIu64        u  print formatting option for uint64_t
PRIuMAX       u  print formatting option for uintmax_t

PRIuFAST8     u  print formatting option for uint_fast8_t
PRIuFAST16    u  print formatting option for uint_fast16_t
PRIuFAST32    u  print formatting option for uint_fast32_t
PRIuFAST64    u  print formatting option for uint_fast64_t
PRIuFAST      u  print formatting option for uintfast_t

PRIuLEAST8    u  print formatting option for uint_least8_t
PRIuLEAST16   u  print formatting option for uint_least16_t
PRIuLEAST32   u  print formatting option for uint_least32_t
PRIuLEAST64   u  print formatting option for uint_least64_t

PRIo8         o  print formatting option for int8_t
PRIo16        o  print formatting option for int16_t
PRIo32        o  print formatting option for int32_t
PRIo64        o  print formatting option for int64_t
PRIoMAX       o  print formatting option for intmax_t

PRIoFAST8     o  print formatting option for int_fast8_t
PRIoFAST16    o  print formatting option for int_fast16_t
PRIoFAST32    o  print formatting option for int_fast32_t
PRIoFAST64    o  print formatting option for int_fast64_t
PRIoFAST      o  print formatting option for intfast_t

PRIoLEAST8    o  print formatting option for int_least8_t
PRIoLEAST16   o  print formatting option for int_least16_t
PRIoLEAST32   o  print formatting option for int_least32_t
PRIoLEAST64   o  print formatting option for int_least64_t

PRIx8         x  print formatting option for int8_t
PRIx16        x  print formatting option for int16_t
PRIx32        x  print formatting option for int32_t
PRIx64        x  print formatting option for int64_t
PRIxMAX       x  print formatting option for intmax_t

PRIxFAST8     x  print formatting option for int_fast8_t
PRIxFAST16    x  print formatting option for int_fast16_t
PRIxFAST32    x  print formatting option for int_fast32_t
PRIxFAST64    x  print formatting option for int_fast64_t
PRIoFAST      x  print formatting option for intfast_t

PRIxLEAST8    x  print formatting option for int_least8_t
PRIxLEAST16   x  print formatting option for int_least16_t
PRIxLEAST32   x  print formatting option for int_least32_t
PRIxLEAST64   x  print formatting option for int_least64_t

PRIX8         X  print formatting option for int8_t
PRIX16        X  print formatting option for int16_t
PRIX32        X  print formatting option for int32_t
PRIX64        X  print formatting option for int64_t

PRIXFAST8     X  print formatting option for int_fast8_t
PRIXFAST16    X  print formatting option for int_fast16_t
PRIXFAST32    X  print formatting option for int_fast32_t
PRIXFAST64    X  print formatting option for int_fast64_t

```

```

PRIXLEAST8   X print formatting option for int_least8_t
PRIXLEAST16  X print formatting option for int_least16_t
PRIXLEAST32  X print formatting option for int_least32_t
PRIXLEAST64  X print formatting option for int_least64_t

```

The following macros can be used as formatting options with the `scanf()` family of functions (see *scanf(3S)*). These macros are used to select the correct formatting option for the integer data types defined earlier in this manual page.

```

SCNd16       d scan formatting option for int16_t
SCNd32       d scan formatting option for int32_t
SCNd64       d scan formatting option for int64_t
SCNdMAX      d scan formatting option for intmax_t

SCNi16       i scan formatting option for int16_t
SCNi32       i scan formatting option for int32_t
SCNi64       i scan formatting option for int64_t
SCNiMAX      i scan formatting option for intmax_t

SCNu16       u scan formatting option for uint16_t
SCNu32       u scan formatting option for uint32_t
SCNu64       u scan formatting option for uint64_t

SCNo16       o scan formatting option for int16_t
SCNo32       o scan formatting option for int32_t
SCNo64       o scan formatting option for int64_t
SCNoMAX      o scan formatting option for intmax_t

SCNx16       x scan formatting option for int16_t
SCNx32       x scan formatting option for int32_t
SCNx64       x scan formatting option for int64_t
SCNxMAX      x scan formatting option for intmax_t

SCNdFAST     d scan formatting option for intfast_t
SCNiFAST     i scan formatting option for intfast_t
SCNoFAST     o scan formatting option for intfast_t
SCNxFAST     x scan formatting option for intfast_t

```

NOTES

The formatting options for the `printf()` family of functions all begin with `PRI`, whereas the formatting options for the `scanf()` family of functions all begin with `SCN`. These formatting strings may not be interchangeable.

EXAMPLES

The following example shows how to use one of the print formatting options with the `printf()` function.

```

uint64_t u;

...
printf("u = %016" PRIx64 "\n", u);

```

AUTHOR

`inttypes.h` was developed by HP.

FILES

`/usr/include/inttypes.h`

SEE ALSO

`printf(3S)`, `scanf(3S)`.

NAME

ioctl - generic device control commands

SYNOPSIS

```
#include <sys/ioctl.h>
ioctl(fildes, request, arg)
int fildes, request;
```

DESCRIPTION

The *ioctl(2)* system call provides for control over open devices. This include file describes *requests* and *arguments* used in *ioctl(2)* which are of a generic nature. For details about how individual requests will affect any particular device, see the corresponding device manual entry in Section (7). If a device does not support an ioctl request it returns **EINVAL**.

FIONREAD

Returns in the long integer whose address is *arg* the number of characters immediately readable from the device file.

FIOSSAIOSTAT

For those character device files which support this command, if the integer whose address is *arg* is non-zero, system asynchronous I/O is enabled; that is, enable SIGIO to be sent to the process currently designated with FIOSSAIOOWN (see below) whenever device-file-dependent events occur. If no process has been designated with FIOSSAIOOWN, then enable SIGIO to be sent to the first process to open the device file.

If the designated process has exited, the SIGIO signal is not sent to any process.

If the integer whose address is *arg* is 0, system asynchronous I/O is disabled.

FIOGSAIOSTAT

For those character device files which support this command, the integer whose address is *arg* is set to 1, if system asynchronous I/O is enabled. Otherwise, the integer whose address is *arg* is set to 0.

FIOSSAIOOWN

For those character device files which support this command, set process ID to receive the SIGIO signals with system asynchronous I/O to the value of the integer whose address is *arg*. Users with appropriate privileges can designate that any process receive the SIGIO signals. If the request is not made by the super-user, only the calling process is allowed to designate that itself or another process whose real or saved effective user ID matches its real or effective user ID, or a process which is a descendant of the calling process, receive the SIGIO signals. If no process can be found corresponding to that specified by the integer whose address is *arg*, the call will fail, with **errno** set to ESRCH. If the request is not made by the super-user and the calling process attempts to designate a process other than itself or (1) another process whose real or saved effective user ID matches its real or effective user ID, or (2) a process which is not a descendant of the calling process, the call fails, with **errno** set to EPERM.

If the designated process subsequently exits, the SIGIO signal will not be sent to any process.

The default when opening a device file is that the process performing the open is set to receive the SIGIO signals.

FIOGSAIOOWN

For those character device files which support this command, the integer whose address is *arg* is set to the process ID designated to receive SIGIO signals.

FIOSNBIO

For those character device files which support this command, if the integer whose address is *arg* is non-zero, non-blocking I/O is enabled; that is, subsequent reads and writes to the device file are handled in a non-blocking manner (see below). If the integer whose address is *arg* is 0, non-blocking I/O is disabled.

For reads, non-blocking I/O prevents all read requests to that device from blocking, whether the requests succeed or fail. Such read requests complete in one of three ways:

- If there is enough data available to satisfy the entire request, the read completes successfully, having read all of the data, and returns the number of bytes read;

- If there is not enough data available to satisfy the entire request, the read completes successfully, having read as much data as possible, and returns the number of bytes it was able to read;
- If there is no data available, the read fails and **errno** is set to EWOULDBLOCK.

For writes, non-blocking I/O prevents all write requests to that device file from blocking, whether the requests succeed or fail. Such a write request completes in one of three ways:

- If there is enough space available in the system to buffer all the data, the write completes successfully, having written out all of the data, and returns the number of bytes written;
- If there is not enough space in the buffer to write out the entire request, the write completes successfully, having written as much data as possible, and returns the number of bytes it was able to write;
- If there is no space in the buffer, the write fails and **errno** is set to EWOULDBLOCK.

To prohibit non-blocking I/O from interfering with the O_NDELAY flag (see *open(2)* and *fcntl(2)*), the functionality of O_NDELAY always supersedes the functionality of non-blocking I/O. This means that if O_NDELAY is set, the driver performs read requests in accordance with the definition of O_NDELAY. When O_NDELAY is not set, the definition of non-blocking I/O applies.

The default on open of a device file is that non-blocking I/O is disabled.

FIOGNBIO

For those character device files which support this command, the integer whose address is *arg* is set to 1, if non-blocking I/O is enabled. Otherwise, the integer whose address is *arg* is set to 0.

WARNINGS

FIOSSAIOSTAT is similar to 4.2 BSD FIOASYNC, with the addition of provisions for security. FIOGSAIOSTAT is of HP origin, complements FIOSSAIOSTAT, and allows saving and restoring system asynchronous I/O TTY state for BSD style job control. FIOSSAIOOWN is similar to 4.2 BSD FIOSETOWN, with the addition of provisions for security. FIOGSAIOOWN is similar to 4.2 BSD FIOGETOWN. Note also the difference that the 4.2 BSD version of this functionality used process groups, while the HP-UX version only uses processes. FIOSNBIO is the same as 4.2 BSD FIONBIO, except that it does not interfere with the AT&T O_NDELAY *open* and *fcntl* flag. FIOGNBIO is of HP origin, complements FIOSNBIO, and allows saving and restoring non-blocking I/O TTY state for BSD-style job control.

SEE ALSO

ioctl(2), socket(7), arp(7).
Section 7 of this manual.

NAME

lang - description of supported languages

DESCRIPTION

HP-UX NLS (Native Language Support) provides support for the processing and customs requirements of a variety of languages. To enable NLS support for a particular language, a language definition must exist on the HP-UX system. Invoking the command **locale -a** (see *locale(1)*) displays information regarding which languages are currently supported on a particular HP-UX system.

The default processing language for HP-UX is **POSIX**. **POSIX** provides an environment in which processing occurs without NLS functionality. This environment is based on the 7-bit-coded USASCII character set.

POSIX and **C** are equivalent and can be used interchangeably.

AUTHOR

lang was developed by HP.

SEE ALSO

locale(1), *localedef(1M)*, *setlocale(3C)*, *environ(5)*, *hpnls(5)*.

NAME

langinfo - language information constants

SYNOPSIS

```
#include <langinfo.h>
```

DESCRIPTION

This header file contains the constants used to identify items of **langinfo** data (see *nl_langinfo(3C)*). The mode of *items* is given in **<nl_types.h>**. The following constants are defined. (**Category** indicates in which *setlocale(3C)* category each item is defined).

Constant	Category	Description
CODESET	LC_CTYPE	Codeset name, such as iso88591 and eucJP .
D_T_FMT	LC_TIME	String for formatting the %c (date and time) directive of <i>date(1)</i> , <i>getdate(3C)</i> , and <i>strftime(3C)</i> .
D_FMT	LC_TIME	String for formatting the %x (date) directive of <i>date(1)</i> , <i>getdate(3C)</i> , and <i>strftime(3C)</i> .
T_FMT	LC_TIME	String for formatting the %X (time) directive of <i>date(1)</i> , <i>getdate(3C)</i> , and <i>strftime(3C)</i> .
T_FMT_AMPM	LC_TIME	Time representation in the 12 hour clock format with AM_STR and PM_STR .
AM_STR	LC_TIME	Ante meridiem string used with 12 hour time formats (AM in English).
PM_STR	LC_TIME	Post meridiem string used with 12 hour time formats (PM in English).
DAY_1	LC_TIME	Name of the first day of the week (Sunday in English).
...
DAY_7	LC_TIME	Name of the seventh day of the week.
ABDAY_1	LC_TIME	Abbreviated name of the first day of the week (Sun in English).
...
ABDAY_7	LC_TIME	Abbreviated name of the seventh day of the week.
MON_1	LC_TIME	Name of the first month in the Gregorian year.
...
MON_12	LC_TIME	Name of the twelfth month.
ABMON_1	LC_TIME	Abbreviated name of the first month.
...
ABMON_12	LC_TIME	Abbreviated name of the twelfth month.
ERA	LC_TIME	The era description segments, which describe how years are counted and displayed for each era in a locale. Each era description segment has the format:

direction: *offset*: *start_date*:
end_date: *era_name*: *era_format*

according to the descriptions below. There will be as many era description segments as are necessary to describe the different eras. Era description segments are separated by semicolons.

Note that the start of an era might not be the earliest point in the era; it might be the latest. For example, the Christian era BC starts on the day before January 1, AD 1, and increases with earlier time.

direction: Either a + or a - character. The + character indicates that years closer to the **start_date** have lower numbers than those closer to the **end_date**.

offset: The number of the year closest to the **start_date** in the era.

start_date: A date in the format *yyyy/mm/dd*, where *yyyy*, *mm*, *dd* are the year, month, and day numbers respectively of the start of the era. Years prior to AD 1 are represented as negative numbers.

end_date: The ending date of the era, in the same format as the *start_date* or one of the two special values *-** or *++*. The value *-** indicates that the ending date is the beginning of time. The value *++* indicates that the ending date is the end of time.

era_name: The era, corresponding to the *%EC* conversion specification.

era_format: The format of the year in the era, corresponding to the *%EY* conversion specification.

ERA_D_FMT	LC_TIME	Format string for formatting the <i>%E</i> (Emperor/Era name and year) directive of <i>date(1)</i> and <i>strftime(3C)</i> if an individual era format is not specified for an era (see <i>localedef(1M)</i>).
ERA_D_T_FMT	LC_TIME	The locale's appropriate alternative date and time format, corresponding to the <i>%Ec</i> field descriptor.
ERA_T_FMT	LC_TIME	The locale's appropriate alternative date and time format, corresponding to the <i>%EX</i> field descriptor.
ALT_DIGITS	LC_NUMERIC	The alternative symbols for digits, corresponding to the <i>%O</i> conversion specification modifier. The value consists of semicolon separated strings. The first string is the alternative symbol corresponding with zero, the second string corresponding with one, etc. Up to 100 alternate symbol strings may be specified.
RADIXCHAR	LC_NUMERIC	Radix character ("decimal point" in English). The string returned is the same as the decimal_point element in the structure returned by <i>localeconv(3C)</i> .
THOUSEP	LC_NUMERIC	Separator for thousands. The string returned is the same as the thousands_sep element in the structure returned by <i>localeconv(3C)</i> .
YESEXPR	LC_MESSAGES	Affirmative response extended regular expression.
NOEXPR	LC_MESSAGES	Negative response extended regular expression.
YESSTR	LC_MESSAGES	Affirmative response for yes/no questions. (Obsolete: use YESEXPR .)
NOSTR	LC_MESSAGES	Negative response for yes/no questions. (Obsolete: use NOEXPR .)
CRNCYSTR	LC_MONETARY	Symbol for currency preceded by <i>-</i> if it precedes the number, <i>+</i> if it follows the number, and <i>.</i> if it replaces the radix. For example, -DM would be used for de_DE.iso88591 (DM1234,56) , + Kr for da_DK.iso88591 (1234,56 Kr) , and .\$ for pt_PT.iso88591 (1234\$56) . See <i>localeconv(3C)</i> for alternative currency formatting information.
DIRECTION	LC_CTYPE	Value to indicate text direction. Values currently defined include null , 0 , and 1 . Values of null or 0 indicate that characters are arranged from left to right within a line and lines are arranged from top to bottom. A value of 1 indicates that characters are arranged from right to left within a line and lines are arranged from top to bottom. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
CONTEXT	LC_CTYPE	String indicating character context analysis. String null or 0 indicates no context analysis is required. String 1 indicates Arabic context analysis required.

ALT_DIGIT	LC_NUMERIC	A string of the characters that are mapped into the ASCII equivalent string <code>0123456789b+-.,eE</code> (where <i>b</i> is a blank). This is also the reverse mapping for output. It is not assumed that the character code values of digits are contiguous or that they are one byte values. A null value for the string indicates that the language has no alternative digits. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
ALT_PUNCT	LC_CTYPE	A string of the characters that are mapped into the ASCII equivalent string <code>b!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~</code> (where <i>b</i> is a blank) in American usage. This is also the reverse mapping for output. It is not assumed that the character code values of punctuation characters are contiguous or that they are one byte values. If any punctuation characters do not have equivalent alternatives, ASCII codes are used in the alternative punctuation string. A null value for the string indicates that the language has no alternative punctuation characters. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
YEAR_UNIT	LC_TIME	Symbol for year. This is usually required to specify date for Asian languages. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
MON_UNIT	LC_TIME	Symbol for month. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
DAY_UNIT	LC_TIME	Symbol for day. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
HOUR_UNIT	LC_TIME	Symbol for hour. This is usually required to specify time for Asian languages. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
MIN_UNIT	LC_TIME	Symbol for minute. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
SEC_UNIT	LC_TIME	Symbol for second. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)
CHARMAP	LC_COLLATE LC_CTYPE	The name of the charmap used to compile this locale. (This constant is an HP proprietary item, subject to change, and may not be portable to other platforms.)

WARNINGS

It is recommended that you use `strftime()` to access date and time information defined in `category` (see `strftime(3C)`), `LC_TIME` and `localeconv(3C)` to access information corresponding to `RADIXCHAR`, `THOUSEP`, and `CRNCYSTR` (see `localeconv(3C)`).

AUTHOR

`langinfo` was developed by HP.

SEE ALSO

`date(1)`, `localedef(1M)`, `getdate(3C)`, `localeconv(3C)`, `nl_langinfo(3C)`, `setlocale(3C)`, `strftime(3C)`, `hpnls(5)`, `lang(5)`.

NAME

libcrash - crash dump access library

SYNOPSIS

```
#include <libcrash.h>

int      cr_open(const char *path, CRASH **cb, int flags);
int      cr_verify(CRASH *crash_cb, int flags);
cr_info_t *cr_info(CRASH *crash_cb);
int      cr_uncompress(CRASH *crash_cb, const char *pathname,
                      uint64_t size, int flags);
int      cr_isaddr(CRASH *crash_cb, uint64_t pagenum, int *avail);
int      cr_read(CRASH *crash_cb, void *buf, uint64_t mem_page,
                int *num_pages);
void     cr_perror(CRASH *crash_cb, int error);
int      cr_close(CRASH *crash_cb);
```

DESCRIPTION

libcrash is a library which provides access to system crash dumps. Access to a dump through the library is independent of the format of the crash dump (there are several, described below). It is also independent of the location of the dump, which could be on a raw dump device, in files in a file system, or a mixture of the two. The memory of a running system can also be treated as a "dump" through use of the `/dev/mem` driver.

All accesses to a dump through the library begin with a call to `cr_open()`. The crash dump descriptor returned from this call is a necessary parameter to all of the other **libcrash** calls. They are:

<code>cr_verify()</code>	Verifies the integrity of a dump by checking the sizes and checksums of all of the files making up the dump.
<code>cr_info()</code>	Returns a pointer to a structure containing information about the dump and the machine and kernel that produced it.
<code>cr_uncompress()</code>	Prepares a file in the crash dump for use, by uncompressing it (if needed) and validating its size and checksum. This function is used internally by the library for access to the physical memory image, and can be used by callers for access to the kernel and kernel module files.
<code>cr_isaddr()</code>	Gives information about whether a particular physical memory page was valid on the machine that dumped, and if so, whether or not that page's contents were included in the dump.
<code>cr_read()</code>	Reads pages from the dump.
<code>cr_perror()</code>	Prints to standard error an error or warning message corresponding to one of the error or warning codes returned by another library call.
<code>cr_close()</code>	Terminates access to the crash dump and frees all space allocated by the library.

Each of the above calls has its own manual page, describing its usage more fully.

Crash Dump Formats

There are three current formats of system crash dumps:

COREFILE	(Version 0) This format, used up through HP-UX 10.01, consists of a single file containing the physical memory image, with a 1-to-1 correspondence between file offset and memory address. Usually there is an associated file containing the kernel image.
COREDUMP	(Version 1) This format, used in HP-UX 10.10, 10.20, and 10.30, consists of a <code>core.n</code> directory containing an <code>INDEX</code> file, the kernel (<code>vmunix</code>) file, and numerous <code>core.n.m</code> files, which contain portions of the physical memory image.
CRASHDIR CURRENT	(Version 2 — the current version) This format, used in HP-UX 11.00 and later, consists of a <code>crash.n</code> directory containing an <code>INDEX</code> file, the kernel and all dynamically loaded kernel module files, and numerous <code>image.m.p</code> files, each of which contain portions of the physical memory image and metadata describing which memory pages were dumped and which were not.

Other formats, for example tape archival formats, may be added in the future.

RETURN VALUES

Most of the calls in **libcrash** return an integer status value. A zero return value indicates success. A positive return value indicates some sort of warning, despite which the requested operation was completed. A negative return value indicates some sort of error, which prevented completion of the requested operation. The values returned by the library are:

CR_OK	Success.
CRWARN_NOEXPECTED	The expected size or checksum of one or more files in the crash dump was not recorded, so the integrity of the dump cannot be verified. The dump might be corrupt.
CRWARN_NOACTUAL	The checksum of one or more files in the crash dump could not be computed, so it could not be checked against the expected value. The dump might be corrupt.
CRWARN_SWAPPEDON	A raw device containing a portion of the crash dump has been swapped on, so the dump is probably corrupt.
CRWARN_MISMATCH	The size or checksum of one or more files in the crash dump did not match what was expected. The dump is probably corrupt.
CRERR_NOPAGE	A read or write request was issued for a memory address that does not exist on the target machine.
CRERR_READONLY	A write request was issued for a crash dump. Writes are supported only to running systems through the /dev/mem driver.
CRERR_WRONGDUMP	A raw dump device which is supposed to contain part of the dump does not. It may have been overwritten by swap activity or by a more recent dump.
CRERR_WRONGHOST	A portion of the crash dump still resides on a dump device of the system that dumped, which is not the current system.
CRERR_ERRNO	A system error occurred. Consult errno for the specific error. Note that certain values for errno have specific meanings in the context of the library. They include:
[ENOEXEC]	A portion of the crash dump could not be uncompressed.
[ENOTDIR]	The specified pathname for the dump was neither a plain file, nor a directory containing an INDEX file, nor the /dev/mem pseudodriver.
\	Other values of errno have their traditional meanings.

AUTHOR

libcrash was developed by HP.

SEE ALSO

cr_close(3), **cr_info(3)**, **cr_isaddr(3)**, **cr_open(3)**, **cr_perror(3)**, **cr_read(3)**, **cr_uncompress(3)**, **cr_verify(3)**.

NAME

limits - implementation-specific constants

SYNOPSIS

#include <limits.h>

DESCRIPTION

The following symbols are defined in <limits.h> and are used throughout the descriptive text of this manual. The column headed **HP-UX Value** lists the values that application writers should assume for portability across all HP-UX systems.

Symbols after values are interpreted as follows:

- + Actual limit might be greater than specified value on certain HP-UX systems.
- Actual limit might be less than the specified value on certain HP-UX systems.
- = Actual limit is always equal to the specified value and does not vary across HP-UX systems.
- * The name of this limit is defined *only* if the preprocessor macro `_XPG2` is defined, either by the compilation flag `-D_XPG2`, or by a `#define` directive in the source before <limits.h> is included in the source.
- # The value defined for this limit might not be a compile-time constant. The value defined always evaluates to an integer expression at run time.

Some of these limits vary with system configuration, and can be determined dynamically by using `sysconf(2)`. Others can vary according to file system or device associated with a specific file, and can be determined with `pathconf(2)`. Others are obsolescent because they are redundant with other limits or not useful in portable applications. They are provided only for importability of applications from other systems, to support applications that comply with the *X/Open Portability Guide, Issue 2*, and for backward compatibility with earlier versions of HP-UX. The `_XPG2` flag should not be defined in new applications.

By including the <limits.h> file in the compilation an application can test the appropriate limits to determine whether it can operate on a particular system, or it might even alter its behavior to match the system to increase its portability across a varying range of limit settings and systems.

Constant	Description	HP-UX Value
ARG_MAX	Max length of arguments to <code>exec(2)</code> in bytes, including environment data	5120 +*
CHAR_BIT	Number of bits in a char	8 =
CHAR_MAX	Max integer value of a char	127 =
CHAR_MIN	Min integer value of a char	-128 =
CHILD_MAX	Max number of simultaneous processes per user ID	25 +-*
CLK_TCK	Number of clock ticks per second	50 +#
DBL_DIG	Digits of precision of a double	15 +
DBL_MAX	Max positive value of a double	1.7976931348623157e+308 +
DBL_MIN	Min positive value of a double	4.94065645841246544e-324 -
FCHR_MAX	Max file offset in bytes	INT_MAX +-*
FLT_DIG	Digits of precision of a float	6 +
FLT_MAX	Max positive value of a float	3.40282346638528860e+38 +
FLT_MIN	Min positive value of a float	1.40129846432481707e-45 -
INT_MAX	Max decimal value of an int	2147483647 +
INT_MIN	Min decimal value of an int	-2147483648 -
LINE_MAX	Max number of characters in a single line	2048 =
LINK_MAX	Max number of links to a single file	32767 +*
LOCK_MAX	Max number of entries in system lock table	32 +-*
LONG_BIT	Number of bits in a long	32 +
LONG_MAX	Max decimal value of a long	2147483647 +
LONG_MIN	Min decimal value of a long	-2147483648 -
MAX_CANON	Max number of bytes in terminal canonical input line	512 +*
MAX_CHAR	Max number of bytes in terminal input queue	MAX_INPUT =*
MAX_INPUT	Max number of bytes in terminal input queue	512 +*

NAME_MAX	Max number of bytes in a path name component	14 +*
NL_ARGMAX	Max value of "digits" in calls to the NLS <i>printf</i> (3S) and <i>scanf</i> (3S) functions	9 =
NL_MSGMAX	Max message number in an NLS message catalog	32767 +
NL_SETMAX	Max set number in an NLS message catalog	255 +
NL_TEXTMAX	Max number of bytes in an NLS message string	8192 +
NGROUPS_MAX	Max number of supplementary groups per process	20 +
OPEN_MAX	Max number of files a process can have open	60 +*
PASS_MAX	Max number of chars in a password	8 +
PATH_MAX	Max number of characters in a path name excluding the null terminator	1023 +*
PID_MAX	Max value for a process ID	30000 +
PIPE_BUF	Max number of bytes atomic in write to a pipe	8192 +*
PIPE_MAX	Max number of bytes writable to a pipe in one write	INT_MAX +
PROC_MAX	Max number of simultaneous processes on system	84 +*
SCHAR_MAX	Max integer value of a signed char	127 =
SCHAR_MIN	Min integer value of a signed char	-128 =
SHRT_MAX	Max decimal value of a short	32767 +
SHRT_MIN	Min decimal value of a short	-32768 -
STD_BLK	Number of bytes in a physical I/O block	512 +
SYSPID_MAX	Max process ID of system processes	4 +*
SYS_NMLN	Length of strings returned by <i>uname</i> (2)	8 +*
SYS_OPEN	Max number of files open on system	120 +*
TMP_MAX	Max number of unique names generated by <i>tmpnam</i> (3S)	17576 +
UCHAR_MAX	Max integer value of an unsigned char	255 =
UID_MAX	Smallest unattainable value for a user or group ID	2147483647 +
UINT_MAX	Max decimal value of an unsigned int	4294967295 +
ULONG_MAX	Max decimal value of an unsigned long	4294967295 +
USHRT_MAX	Max decimal value of an unsigned short	65535 +
USI_MAX	Max decimal value of an unsigned int	UINT_MAX =*
WORD_BIT	Number of bits in a "word" (int)	32 +

EXAMPLES

UID_MAX has an HP-UX value of **2147483647 +**, which means that on all HP-UX systems the smallest unattainable value for a user or group ID is at least 2147483647. A particular system might be capable of supporting more than 2147483647 user or group IDs, in which case its **<limits.h>** file sets **UID_MAX** to a higher value; however, any application assuming such a higher value is not guaranteed to be portable to all HP-UX systems.

AUTHOR

limits was developed by HP.

SEE ALSO

exec(2), *fcntl*(2), *fork*(2), *getgroups*(2), *link*(2), *lockf*(2), *open*(2), *pathconf*(2), *sysconf*(2), *uname*(2), *write*(2), *printf*(3S), *scanf*(3S), *tmpnam*(3S), *passwd*(4), *values*(5), *termio*(7).

Series 700 config(1M).

STANDARDS CONFORMANCE

<limits.h>: AES, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1, POSIX.2, ANSI C

NAME

man - macros for formatting manpages

SYNOPSIS

man *file* ...

nroff **-man** [*option*]... [*file*]...

DESCRIPTION

The **man** macros are used by the **man** and **nroff** commands (see *man(1)* and *nroff(1)*) — and are usable by **troff** (see third-party documentation) — to format the on-line versions of manpages found in *HP-UX Reference* and other related reference manuals. The **man** command calls **nroff**.

man and nroff Defaults

The default page size is 85 characters by 66 lines (8.5×11 inches), with a 75-character by 60-line text area. Hyphenation is turned off and paragraphs are left-adjusted, ragged-right.

troff Defaults

The default page size is 8.5×11 inches with a 6.5×10-inch text area. The type size is 10 points and the vertical line spacing is 12 points. Hyphenation is turned on and paragraphs are justified left and right.

Other Defaults

Type font and size are reset to default values before each paragraph and after processing font- and size-setting macros such as **.I**, **.RB**, and **.SM**. Tab stops are neither used nor set by any macro except **.DT** and **.TH**. The **.TH** macro invokes **.DT** (see below).

Options

The following options can be specified for **nroff** or **troff**. They are not permitted for the **man** command.

- rs1** Reduce the dimensions for **troff** to a page size of 6×9 inches with a 4.75×8.375-inch text area, the type size to 9 points, and the vertical line spacing to 10 points. This option is ignored by **nroff**.
- rv2** Set certain parameters to values appropriate for certain Versatec printers: line length to 82 characters (ens); page length to 84 lines; underlining inhibited. Do not confuse this option with the **-Tvp** option of the **man** command, which is available on some operating systems, but not on HP-UX.

Summary of Macros, Strings, and Numbers

The defined **man** macros, strings, and numbers are summarized here and described in detail in the following subsections.

- .B** Set *text* in bold.
- .BC** Set *words* alternately in bold and constant-width.
- .BI** Set *words* alternately in bold and italics.
- .BR** Set *words* alternately in bold and roman.
- .C** Set *text* in constant-width.
- .CB** Set *words* alternately in constant-width and bold.
- .CD** Identify command name.
- .CI** Set *words* alternately in constant-width and italics.
- .CR** Set *words* alternately in constant-width and roman.
- .CT** Identify citation title.
- .DT** Restore default tab settings.
- .EM** Identify emphasis.
- .ER** Identify error name.
- .EV** Identify environment variable name.
- .GT** Identify glossary term.
- .HP** Begin paragraph with hanging indent.
- .I** Set *text* in italics.
- .IB** Set *words* alternately in italics and bold.
- .IC** Set *words* alternately in italics and constant-width.
- .IP** Begin indented paragraph with optional tag.

.IR	Set <i>words</i> alternately in italics and roman.
.KC	Identify keycap.
.P	Begin normal paragraph.
.PD	Set interparagraph spacing.
.PM	Use Bell System proprietary subfooters.
.PP	Begin normal paragraph.
.RB	Set <i>words</i> alternately in roman and bold.
.RC	Set <i>words</i> alternately in roman and constant-width.
.RE	End relative margin indent.
.RI	Set <i>words</i> alternately in roman and italics.
.RS	Start relative margin indent.
.RV	Identify return value.
.S3	Insert third level header.
.SC	Identify system constant name.
.SH	Insert section header.
.SM	Print <i>text</i> one point smaller.
.SS	Insert subsection header.
.TH	Start new manpage and define page headers and footers.
.TP	Begin tagged paragraph.
*R	Registered trademark.
*S	Change to default type size.
*(Tm	Trademark.
\n(IN	Left text margin; default margin indent and paragraph indent.
\n(LL	Line length, including \n(IN.
\n(PD	Interparagraph distance.

Macro Parameters

All macro parameters are positional and can be omitted, starting from the right. Each parameter is a *word*, as described below.

<i>mi</i>	Margin increment. This is the amount by which the left text margin will be increased. If it is omitted, it defaults to \n(IN basic units (u). The default measure for <i>mi</i> is ens (n). The left text margin's base value is \n(IN.
<i>in</i>	Paragraph indent. This is the amount by which indented lines of a paragraph will be indented. If it is omitted, it defaults to the value that was established by the most recent paragraph macro: explicitly or default by .HP, .IP, or .TP; implicitly by .P, .PP, .RE, or .RS. The default measure for <i>in</i> is ens (n).
<i>text</i>	Consists of zero to six <i>words</i> . If <i>text</i> is empty, the special treatment is applied to the next line containing text to be printed. For example, .I can be used to italicize a whole line, or .SM followed by .B to make small bold text.
<i>word</i>	A string of characters separated by spaces (not tabs). Use quotation marks ("word") to include spaces in a <i>word</i> ("string string") or to specify a null <i>word</i> (""). (The nonbreaking, nonpaddable space (\) is not a separating space.)

Header Macros

.TH	<i>t1 s2 c3 n4 a5</i>	Set the title and entry heading. <i>t1</i> , <i>s2</i> , <i>c3</i> , <i>n4</i> , and <i>a5</i> are <i>words</i> .
<i>t1</i>		Entry title.
<i>s2</i>		Section number. <i>t1</i> is combined with <i>s2</i> in parentheses to form the top left- and righthand corners of the page heading.
<i>c3</i>		Extra commentary, such as "Optional Software Required". It is placed at the center of the bottom line in the two- or three-line page heading space.
<i>n4</i>		Other notations, such as "Series 300/400 Only". It is centered between the title and section on the first page heading line.
<i>a5</i>		Support for alternate naming, such as a FORTRAN routine name corresponding to a C function name specified in <i>t1</i> .

The resulting output is in the form:

<i>tl(s2)</i>	<i>n4</i>	<i>tl(s2)</i>
<i>a5</i>		<i>a5</i>
	<i>c3</i>	

- .SH** *text* Place section head *text*, such as **SYNOPSIS**, here. Section headings start at the left margin. Since they are normally all uppercase, they are printed a point-size smaller in **troff**.
- .SS** *text* Place subsection head *text*, such as **Options**, here. Subsection headings start between the left margin and the normal text indent.
- .S3** *text* Place third-level head *text*, such as subhead, here. Third-level headings start at the normal text indent.

Paragraph Macros

- .P**
- .PP** Begin a block paragraph. Reset *in* to `\n(IN, "cancelling"` any value set by previous **.HP**, **.IP**, and **.TP** macros.
- .HP** *in* Begin a paragraph with hanging indent. The text begins at the current margin. The second and following output lines are indented.
- .TP** *in* Begin an indented paragraph with hanging tag. The next input line that contains text to be printed is taken as the tag. The tag begins at the current margin. If the tag fits in the indent, the paragraph text begins at the indent position on the same line. If the tag does not fit in the indent, the paragraph text begins at the indent position on the next line.
- .IP** *t in* Same as **.TP** *in* with tag *t*. Often used to get an indented paragraph without a tag.
- .RS** *mi* Increase the current left margin by *mi*. If *mi* is omitted, it defaults to the current value of *in*. Set the paragraph indent, *in*, for the new margin level to `\n(IN`. You can specify up to nine **.RS** increment levels. Margin increments can be backed out with the **.RE** macro and are reset to the base margin by the **.TH**, **.SH**, **.SS**, and **.S3** header macros.
- .RE** *k* Return to the *k*th left margin setting (initially, *k*=1; *k*=0 is equivalent to *k*=1). If *k* is omitted, return to the previous margin value. The paragraph indent, *in*, is restored to the value it had prior to the corresponding **.RS** macro.
- .PD** *pd* Set the interparagraph distance to *pd* vertical spaces. If *pd* is omitted, set the interparagraph distance to the default value: 1 line in **nroff**. 0.4 line in **troff**. The measure for *pd* is vertical line spaces (**v**).

Font Macros

- .B** *text* Set *text* in bold.
- .C** *text* Set *text* in constant-width font. See the WARNINGS section.
- .I** *text* Set *text* in italic.

(There is no **.R** (roman) macro, but you can use one of the **.XY** combinations if need be.)

- .XY** *a b* Concatenate *a* in font *X* with *b* in font *Y* and alternate these two fonts for up to six words. The font letters *X* and *Y* can be **B** (bold), **C** (constant-width), **I** (italic), and **R** (roman), in the following combinations:

	.CB	.IB	.RB
.BC		.IC	.RC
.BI	.CI		.RI
.BR	.CR	.IR	

For more about constant-width font, see the WARNINGS section.

- .SM** *text* Make *text* one point smaller than the default point size. This has no effect in **nroff**.

Special Macros

These macros identify common text elements in manpages. They aid in providing consistent font usage in the HP style and in improving conversion to other formatting systems.

The first parameter is set in an appropriate font or format. The second parameter, *punctuation*, is set in the roman font and is provided for concatenated punctuation. The two parameters are concatenated as with the font macros.

- .CD *commandname punctuation*
commandname is a command name, usually defined in a section 1 or 1M manpage, such as **man**. It is displayed in constant-width font.
- .CT *citationtitle punctuation*
citationtitle is the name of a document, such as *HP-UX Reference*. It is displayed in italics. (Use the standard .IR macro for manpage references.)
- .EM *emphasis punctuation*
emphasis is a word or phrase that you want to emphasize, such as *Do not*. Use emphasis sparingly. It is displayed in italics. (Use the standard .I macro for variable names.)
- .ER *errorname punctuation*
errorname is an error name that corresponds to a value assigned to **errno** by a function and described in the ERRORS section of a manpage. It is displayed in roman, enclosed in square brackets. For example, .ER EIO . is displayed as [EIO].
- .EV *environvarname punctuation*
environvarname is the name of an environmental variable, such as **PATH**. It is displayed in constant-width font.
- .GT *glossterm punctuation*
glossterm is a glossary term, or a term that you are defining for later use in the manpage, such as **path name**. It is displayed in bold.
- .KC *keycap punctuation*
keycap is the name of a keyboard key, such as **Tab**. It is displayed in bold.
- .RV *returnvalue punctuation*
returnvalue is a numerical return value of a function or exit status of a command, usually as defined in a RETURN VALUE or EXIT STATUS section. It is generally used for number expressions, such as 0, >3, and <>0, but not for word descriptions, such as "nonzero". It is displayed in constant-width font.
- .SC *systemconstant punctuation*
systemconstant is an operating system constant name, such as **PATH_MAX**. It is displayed in constant-width font. See *getconf(1)*.

Other Macros

- .DT Restore default tab settings: every 5 ens in **nroff**; every 3.6 ems in **troff**.
- .PM *sf* Produce Bell System proprietary subfooters.
sf produces
P PRIVATE
N NOTICE
BP BELL LABORATORIES PROPRIETARY
BR BELL LABORATORIES RESTRICTED

Strings

The following string references are defined:

- *R Registered Trademark symbol: displayed as (**Reg.**) in **nroff**, and using the \(\rg inline macro in **troff**.
- *S Change to default type size. This is executed as a \s inline macro.
- *(Tm Trademark indicator, TM, displayed as a superscript if possible.

Numbers

The following number references are defined:

- \n(IN Left text margin indent relative to section heads and default value for *mi* and *in*: 5 ens in **nroff**; 3.6 ems in **troff**. IN is expressed in basic units (u).

- `\n(LL` Line length including `\n(IN`: 75 characters in `nroff`; 6.5 inches in `troff`. Also see the Options subsection. `LL` is expressed in basic units (`u`).
- `\n(PD` Current interparagraph distance. Set by `.PD`. `PD` is expressed in vertical line spaces (`v`).

Measure

`nroff` and `troff` use a number of scale indicators to qualify horizontal and vertical measurements. Many macro parameters have default units of measure. Because all assignments to numeric variables are converted to basic units (`u`), it is important to take care in assigning and referencing values.

Scale		Basic Units	
Indicator	Measure	nroff	troff
c	centimeter	240/2.54	D/2.54
i	inch	240	D
m	em	C	p*S
n	en = em/2	C	p*S/2
p	point = 1/72 inch	240/72	D/72
P	pica = 1/6 inch	240/6	D/6
u	basic unit	1	1
v	vertical line space	V	V

- C Character width of output device.
- D Dots per inch (dpi) of output device.
- S Current type size in points.
- V Current vertical line spacing in basic units.

Font Conventions

Entries in the *HP-UX Reference* use the following font conventions:

- roman Regular typeface.
- italic Used for variables and other words that represent an argument that may take on a user-defined or variable value. Also used for *emphasis*.
- boldface Used primarily in headings and occasionally for terms when first introduced or when being defined.
- constant-width Used for all literals that are typed exactly as shown when used as keyboard commands or command-line options, in programs, etc.

Page Footers

The strings used in the page footer are initialized by the `.TH` macro. `)H` defaults to a nonprinting (null) string. `JW` defaults to the date the manpage is formatted, as in **Formatted: July 5, 1995**. In HP manpages, `)H` is set to the company name, "Hewlett-Packard Company"; `JW` is set to release information, as in "HP-UX Release 10.10: November 1995".

This arrangement enables users and third-party software suppliers to directly control the contents of the left- and right-hand fields of the footer line for use in displaying company name, release version, etc., as desired when creating their own manual entries. `)H` is printed on the left, `JW` is printed on the right, and the page number is printed in the center. These strings can be defined anywhere after the `.TH` macro call, provided they appear before the end of the first page. For example, the following source file segment:

```
.TH man 5
.ds )H XYZ Company
.ds JW Release 2.3: May 1996
```

produces a footer with text in the form:

XYZ Company	- 1 -	Release 2.3: May 1996
-------------	-------	-----------------------

Tables

The `tbl` preprocessor (see *tbl(1)*) can be used to insert tables in manpages. The `man` macros allow you to use the standard `tbl` macros: `.TS`, `.T&`, and `.TE`. They do not support the `mm` macro extensions,



m

.TS **H** and **.TH** (see *mm(5)*).

In general, avoid using the **man** macros within a table, particularly the font macros, which can produce peculiar and unpredictable results. Use the **nroff/troff** intrinsic macros instead. For example, to specify bold type, use the in-line macro **\fB**, or, more generally, **\f3**. To insert horizontal blank lines, you can use the **.PP** or **.IP** macro, depending on which one preceded the table, but you must then avoid using the **center** and **expand** table format specifications. Otherwise, indentation will be erratic.

WARNINGS

HP no longer uses the NAME section to prepare the Table of Contents and Index for the printed manual. Instead, that information is coded as comments at the end of each manpage source file where it can be accessed by various tools and programs as desired. The NAME section is still used for the **whatis** database, as described below.

The macro package used to print the *HP-UX Reference* increases the interword spaces (to eliminate ambiguity) in the **SYNOPSIS** section of each entry.

In addition to the macros, strings, and number registers mentioned above, a number of internal macros, strings, and number registers are defined. These include

- The names predefined by the **nroff/troff** processor.
- The macro **th**,
- The number register **:m**,
- Macro, string, and number names in the forms **)x**, **]x**, and **}x**, where *x* stands for some alphanumeric character,
- Macro, string, and number names in the form *XY*, where *X* and *Y* are capital letters.

Fonts

nroff uses only three fonts: roman, italic, and bold, designated as font positions 1, 2, and 3, respectively. The constant-width font macros in the macro package (**.C**, **.RC**, **.IC**, **.BC**, **.CR**, **.CI**, **.CB**) simulate a constant-width font with the bold font, since all **nroff** output is constant width or typewriter format.

To use a true constant-width font with **troff**, change the corresponding font 3 specification in each constant-width font macro to font 4 and mount a constant-width font in position 4 using a **troff .fp** request, as in:

```
.fp 4 CW
```

The **whatis** Database

The NAME section of each manpage is processed by **catman** (see *catman(1M)*) to create an entry in the **whatis** database, which is used by the **-f** and **-k** options of the **man** command. **catman** processes the lines of the NAME section into a single line in the format:

```
name[, name]... - explanatory-text
```

Hyphens (typed as **-** or **\-**), en-dashes (**\(mi)**), and em-dashes (**\(em)**) are treated equivalently. The last space-hyphen-space in the line becomes the dividing point between names and explanatory text.

FILES

```
/usr/share/lib/macros/an
```

The **man** macros.

```
/usr/share/lib/tmac/tmac.an
```

Called by **man**. Sources (**.so**) the macros in **/usr/share/lib/macros/an**. Other macro files can be specified here to accommodate manpages with additional macro requirements.

```
/usr/share/lib/whatis
```

File of strings from manpage NAME sections, created by **catman**, and used by the **man -k** and **-f** options.

SEE ALSO

col(1), *man(1)*, *neqn(1)*, *nroff(1)*, *tbl(1)*, *catman(1M)*, *mm(5)*.

NAME

manuals - current list of HP-UX documentation

DESCRIPTION

This entry contains a list of current English language manuals for the HP-UX operating system. It also lists Japanese, Korean, and Chinese NLI/O manuals.

The information is current as of HP-UX release 11.0. Manuals updated after release 11.0 may have different part numbers. Contact your HP Sales and Support office for assistance on getting the correct editions of manuals for your system. If discrepancies are encountered between system behavior and a given manual, verify the edition date to ensure that the manual is current.

Applications software manuals are not included in this list.

Ordering Information

For information about how to order any of these manuals, as well as other HP computer and calculator manuals and supplies, call *Parts Direct Ordering* toll-free in the United States at **1-800-227-8164**; or, if outside the U.S.A., contact your nearest HP Sales and Support office for information about ordering these manuals or versions of these manuals in other languages.

Manual List

Stock Number	Title
28604-90001	HP OSI Express 802.4 Hardware
32055-90001	HP X.400/HP Desk Administration Guide
32055-90002	Using Hp DeskMgr Connection X
32070-90030	Installing and Administering OSI Transport Services
32070-90031	HP OSI Troubleshooting Guide
32070-90032	ACSE/Presentation and ROSE Interface Programmer's Guide
32070-90033	Release Notes: OTS 9000 10.0
36960-90050	Programmer's Guide to X.25 Link Software for HP9000 Systems
5963-4146	HP ATM/9000 Adapter for HSC Servers Config & Troubleshooting
5963-4416	C++ Customer Survey
5963-4467	C/HP-UX 10.0 Release Notes
5963-4468	C/HP-UX Technical Addendum
5963-4475	HP-UX Programming Tools Release Notes
5963-4519	FORTTRAN/9000 Release Notes
5963-8917	HP Pascal/HP-UX Release Notes
5963-8942	HP-UX 10.0 File System Layout Whitepaper
5964-1320	HP Micro Focus COBOL Release Notes S700/800
5964-1365	HP-UX 10.10 Release Notes (On-Line and LRom only)
5965-4409	HP-UX Programming Tools Release Notes
5965-4445	HP Fortran 90 Release Notes
5965-4446	C/HP-UX Developer's Bundle Release Notes
5965-4447	HP Micro Focus COBOL Release Notes
5965-4448	Nihongo COBOL Release Notes
5965-4462	JAVA Developer's Kit Release Notes
5965-4463	JAVA Virtual Machine Release Notes
5965-4464	JAVA JIT Release Notes
5965-4465	HP Pascal/HP-UX Release Notes
5965-4629	HP aC++ Customer Survey
5965-4641	HP-UX Memory Management White Paper (replaced at 11.0)
5965-4642	HP-UX Process Management White Paper (replaced at 11.0)
5965-4643	HP-UX Multi-Processing White Paper
5966-8634	ATM/622 & ATM/155 Adapters Config and Troubleshooting Guide
5966-9847	HP C++ 12.0 (cfront) Release Notes
5966-9852	HP aC++ Release Notes
5967-0055	Release Notes for HP-UX 11.0 (On-Line and CD-ROM only)
5967-0056	HP-UX 11.0 Documentation Map White Paper
92431-90005	HP Pascal/HP-UX Reference Manual
92431-90006	HP Pascal/HP-UX Programmer Guide
92432-90011	Assembly Language Reference Manual
92434-90002	HP C Programmer's Guide

92453-90024	HP C/HP-UX Reference Manual
92552-90009	NLIO Input Method Guide (Trad. Chinese)
92668-90006	Managing SwitchOver/UX
92668-90007	Read Me Before Upgrading SwitchOver/UX from HP-UX 9.0 to HP-UX 10.0
97005-90015	The Ultimate Guide to vi and ex Editors
98194-90053	LLA Programming & Migration Guide
A4011-90003	Token Ring Series 712 Quick Installation
A4011-90006	Installing and Administering Token Ring
B1011-90003	Using LAN Manager/X
B1014-90012	DTC Device File Access Utilities and Telnet Port Identification
B1031-90002	Installing & Administering NFS Services
B1033-90004	HP FTAM/9000 Reference Manual
B1033-90014	HP FTAM/9000 Programmer's Guide
B1033-90024	HP FTAM/9000 User's Guide
B1033-90034	Installing and Administering HP FTAM/9000
B1033-90040	FTAM/9000 Technical Addendum
B1171-90078	HP Xlib Extensions (CD-ROM only)
B1171-90101	CDE 1.0 User's Guide
B1171-90102	CDE 1.0 Advanced User's and System Administrator's Guide
B1171-90104	HP CDE Getting Started Guide
B1171-90105	CDE Programmer's Overview
B1171-90106	CDE Programmer's Guide
B1171-90108	CDE Desktop Korn Shell User's Guide
B1171-90109	CDE Help System Author's and Programmer's Guide
B1171-90126	CDE ToolTalk Programmer's Guide
B1171-90135	HP CDE Getting Started Guide Addendum for 10.30
B1171-90136	Technical Print Service System Administration's Guide (CD-ROM only)
B1171-90141	HP CDE Getting Started Guide
B1171-90145	Motif Programmer's Guide Release 2.1 (CD-ROM only)
B1171-90146	Motif User's Guide Release 2.1 (CD-ROM only)
B1171-90147	Motif and CDE Style Guide Reference (CD-ROM only)
B1171-90148	Motif and CDE Style Guide (CD-ROM only)
B1171-90149	Motif and CDE Style Guide and Certification Checklist (CD-ROM only)
B1171-90150	Motif Widget Writer's Guide (CD-ROM only)
B1171-90151	Motif Master Glossary (CD-ROM only)
B1171-90152	CDE Programmer's Overview (CD-ROM only)
B1171-90153	CDE Programmer's Guide (CD-ROM only)
B1171-90154	CDE Help System Author's and Programmer's Guide (CD-ROM only)
B1171-90155	CDE Internationalization Programmer's Guide (CD-ROM only)
B1171-90156	CDE Desktop Korn Shell User's Guide (CD-ROM only)
B1171-90157	CDE ToolTalk Messaging Overview (CD-ROM only)
B1171-90158	Guide to the DocBook DTD (CD-ROM only)
B1171-90159	CDE Product Glossary (CD-ROM only)
B1171-90161	CDE 2.1 User's Guide (CD-ROM only)
B1171-90162	CDE 2.1 Advanced User's and System Administrator's Guide (CD-ROM only)
B1171-90163	CDE ToolTalk Programmer's Guide (CD-ROM only)
B1171-95101	CDE1.0 Users Guide (French)
B1171-95102	CDE1.0 System Administration Guide (French)
B1171-95141	CDE2.1 Getting Started (French)
B1171-96101	CDE1.0 Users Guide (German)
B1171-96102	CDE1.0 System Administration Guide (German)
B1171-96141	CDE2.1 Getting Started (German)
B1171-97101	CDE1.0 Users Guide (Spanish)
B1171-97102	CDE1.0 System Administration Guide (Spanish)
B1171-97141	CDE2.1 Getting Started (Spanish)
B1171-98101	CDE1.0 Users Guide (Italian)
B1171-98102	CDE1.0 System Administration Guide (Italian)
B1171-98141	CDE2.1 Getting Started (Italian)
B1171-99101	CDE1.0 Users Guide (Swedish)
B1171-99102	CDE1.0 System Administration Guide (Swedish)
B1171-99141	CDE2.1 Getting Started (Swedish)
B2200-90019	Japanese NLIO Manual

B2204-90002	NLIO System Admin Guide (Kor.)
B2204-90011	NLIO Access User's Guide (Kor.)
B2208-90011	NLIO Access User's Guide (Trad. Chinese)
B2208-90012	NLIO System Admin Guide (Trad. Chinese)
B2212-90002	NLIO System Admin Guide (Simp. Chinese)
B2212-90003	NLIO Input Method Guide (Simp. Chinese)
B2212-90011	NLIO Access User's Guide (Simp. Chinese)
B2355-90046	Shells: User's Guide
B2355-90049	POSIX Conformance Document
B2355-90060	Programming with Threads on HP-UX
B2355-90063	Kermit Mailer Card
B2355-90068	Using the Image Developer's Kit (CD-ROM only)
B2355-90069	Using the Audio Developer's Kit (CD-ROM only)
B2355-90072	Technical Addendum to the Shells: User's Guide
B2355-90086	Installing HP-UX 10.10 and Updating 10.0x to 10.10
B2355-90087	Installing & Administering LAN/9000
B2355-90088	Using Serial Line IP Printers
B2355-90088	Using Serial Line IP Protocols
B2355-90092	BSD Socket Interface Programmer's Guide
B2355-90093	DLPI Programmer's Guide
B2355-90094	XTI Programmer's Guide
B2355-90095	BSD Socket Quick Reference Guide
B2355-90112	Quick Guide to Installing HP-UX 10.20
B2355-90133	Installing & Administering Internet Services
B2355-90134	Using Internet Services
B2355-90140	Networking Overview
B2355-90153	Installing HP-UX 11.0 and Updating 10.x to 11.0
B2355-90154	Managing HP-UX Software with SD-UX
B2355-90155	HP-UX Distributed Print Service Administration Guide
B2355-90156	HP-UX Distributed Print Service User's Guide
B2355-90157	Managing Systems and Workgroups
B2355-90158	Configuring HP-UX for Peripherals
B2355-90159	HP-UX Reference Vol 1 (11.0)
B2355-90160	HP-UX Reference Vol 2 (11.0)
B2355-90161	HP-UX Reference Vol 3 (11.0)
B2355-90162	HP-UX Reference Vol 4 (11.0)
B2355-90163	HP-UX Reference Vol 5 (11.0)
B2355-90164	Using HP-UX
B2355-90166	HP-UX Reference (5 Volume Customer Order Number)
B2355-90655	HP-UX Linker and Libraries User's Guide
B2355-95153	Installing HP-UX 11.0 & Updating 10.x to 11.0 (French)
B2355-95154	Managing HP-UX Software with SD-UX (French)
B2355-95157	Managing Systems and Workgroups (French)
B2355-95158	Configuring HP-UX for Peripherals (French)
B2355-95164	Using HP-UX (French)
B2355-96153	Installing HP-UX 11.0 & Updating 10.x to 11.0 (German)
B2355-96154	Managing HP-UX Software with SD-UX (German)
B2355-96155	HP-UX Distributed Print System User's Guide (German)
B2355-96156	HP-UX Distributed Print System Administration Guide (German)
B2355-96157	Managing Systems and Workgroups (German)
B2355-96158	Configuring HP-UX for Peripherals (German)
B2355-96164	Using HP-UX (German)
B2433-90029	HP Micro Focus COBOL Language Reference, Additional Topics
B2433-90031	HP Micro Focus COBOL User's Guide
B2433-90056	HP Micro Focus System Reference Vol. #1
B2433-90057	HP Micro Focus System Reference Vol. #2
B2433-90058	HP Micro Focus Language Reference Vol. #1
B2433-90059	HP Micro Focus Language Reference Vol. #2
B2433-90060	HP Micro Focus User's Guide
B2433-90061	HP Micro Focus Getting Started
B2433-90062	HP Micro Focus Compatibility Guide
B2433-90063	HP Micro Focus Error Messages

B3190-90011	DCE Application Environment Specification
B3190-90018	Understanding DCE
B3190-90029	Guide to Writing DCE Applications
B3190-90037	OSF DCE Application Development Reference
B3190-90038	OSF DCE Application Development Guide, I
B3190-90039	OSF DCE Application Development Guide, II
B3190-90040	OSF DCE Application Development Guide, III
B3190-90046	Introduction to OSF DCE
B3190-90046	Introduction to OSF DCE
B3190-90047	OSF DCE Command Reference
B3190-90048	OSF DCE System Administration: Core Components
B3190-90049	OSF DCE DFS Administration Guide & Reference
B3190-90050	OSF DCE GDS Administration Guide & Reference
B3190-90052	Planning and Configuring HP DCE
B3393-90006	Developer's Toolkit Release Notes for HP-UX 10.30
B3476-90017	HP PAK Performance Analysis User's Guide
B3782-90034	Moving HP-UX 9.x Code and Scripts to 10.x (Analysis & Conversion)
B3782-90073	Upgrading from HP-UX 9.x to 10.x
B3782-90074	Read Me Before Installing or Upgrading HP-UX 10.10
B3782-90076	Read Me Before Preparing for Upgrade from HP-UX 9.x to 10.x
B3782-90103	Read Me Before Installing or Updating HP-UX 10.20
B3782-90176	Support Media User's Manual
B3782-90192	Read Me Before Installing or Updating HP-UX 11.00
B3834-90008	HP Process Resource Manager User's Guide
B3906-90001	HP-UX FORTRAN/9000 Programmer's Guide
B3906-90002	HP-UX FORTRAN/9000 Programmer's Reference
B3906-90005	HP-UX Floating-Point Guide
B3928-90001	HP Online JFS Installation Instructions
B3936-90019	Managing MC/ServiceGuard
B5158-90001	Configuring OPS Clusters with MC/LockManager
J2120-62000	Using the HP DTC Manager/UX
J2120-90031	HP DTC Manager/UX: Release Notes
J2157-90013	Installing and Administering FDDI
J2157-90014	FDDI/9000 Quick Installation Guide
J2163-90005	Release Notes: FTAM 9000
J2165-90012	Token Ring S700 10.0 Quick Installation
J2165-90017	EISA Token Ring Release Notes
J2166-90008	Token Ring S800 10.0 Quick Installation
J2399-90000	VT3K Quick Reference
J2399-90001	VT3K Customer Letter
J2496-90000	Using DTC 16RX
J2496-90012	DTC 16RX Release Notes
J2655-90003	EISA 100VG AnyLAN Quick Installation Guide
J2655-90007	Installing and Administering EISA/100VG AnyLAN
J2655-90011	100VG AnyLAN for Series 700 Release Notes
J2669-90005	LMU Installation & Administration
J2669-90006	LMU Troubleshooting and Commands
J2669-90007	LMU 1.x to 2.2 Migration
J2669-90008	Using LMU 2.2
J2671-90008	Netware v3.12 Installation and Administration
J2671-90009	Netware v3.12 Release Notes
J2695-90002	EISA 100VG AnyLAN for Series 800 Release Notes
J2793-90000	X.25/9000 User's Guide
J2793-90034	X.25 J2793A Release Notes
J2805-90001	ATM Release Notes
J2806-90021	ATM Release Notes

For information about the availability of any of these manuals in your local language, contact your sales center.

For HP Micro Focus COBOL manuals, contact your HP sales representative and request the manuals-only option of the product.

Hewlett-Packard also recommends the following retail books:

Clusters for High Availability: A Primer of HP-UX Solutions,
Prentice Hall (ISBN 0-13-494758-4)

Disk and File Management Tasks on HP-UX,
Prentice Hall (ISBN 0-13-518861-X)

Go Solo: How to Implement and Go Solo with the Single UNIX Specification,
Prentice Hall (ISBN 0-13-439381-3)

A Guide to Netware for UNIX,
Prentice Hall (ISBN 0-13-300716-2)

HP-UX 10.X System Administration "How To" Book,
Prentice Hall (ISBN 0-13-125873-7)

Kornshell Programming Tutorial,
Addison Wesley (ISBN 0-201-56324-X)

Learning the HP-UX Operating System,
Prentice Hall (ISBN 0-13-258534-0)

The New Kornshell Command and Programming Language,
Prentice Hall (ISBN 1-13-182700-6)

Portable Shell Programming: An Extensive Collection of Bourne Shell Examples,
Prentice Hall (ISBN 0-13-451494-7)

Practical DCE Programming,
Prentice Hall (ISBN 0-13-324419-9)

sendmail,
O'Reilly & Associates (ISBN 1-56592-222-0)

Thread Time: The MultiThreaded Programming Guide,
Prentice Hall (ISBN 0-13-190067-6)

The UNIX C Shell Field Guide,
Prentice Hall (ISBN 0-13-937468-X)

Using & Managing UUCP,
O'Reilly & Associates (ISBN 1-56592-153-4)

Using C-Kermit,
Digital Press, Columbia University (ISBN 1-55558-164-1)

Volume 3: X Window System User's Guide, Standard Edition,
O'Reilly & Associates (ISBN 1-56592-014-7)

Volume 3M: X Window System User's Guide, Motif Edition,
O'Reilly & Associates (ISBN 1-56592-015-5)

Writing Your Own OSF Motif Widgets,
Prentice Hall (ISBN 0-13-104191-6)

To order the above retail books, call:

Addison Wesley:
1-800-822-6339 (from the U.S)
1-800-447-2226 x5190 (for international orders)

Digital Press:
212-854-3703

O'Reilly and Associates, Inc.:
1-800-998-9938

Prentice Hall:
1-800-947-7700 (from the U.S.)
1-800-922-0579 (for government orders)
201-767-4990 (for international orders)

NAME

math - math functions and constants

SYNOPSIS

```
#include <math.h>
```

DESCRIPTION

This file contains declarations of all the functions in the Math Library (described in Section (3M)).

It also defines the following constants that may be used in initializations or returned as error values:

HUGE_VAL The maximum value (type **double**) of a double-precision floating-point number (IEEE positive INFINITY).

NAN A quiet NaN (Not-a-Number) value (type **float**).

INFINITY A positive infinity value (type **float**).

The following mathematical constants are defined for user convenience:

M_E The base of natural logarithms (e).

M_LOG2E The base-2 logarithm of e .

M_LOG10E The base-10 logarithm of e .

M_LN2 The natural logarithm of 2.

M_LN10 The natural logarithm of 10.

M_PI The ratio of the circumference of a circle to its diameter. (There are also several fractions of π , its reciprocal, and its square root: **M_PI_2**, **M_PI_4**, **M_1_PI**, **M_2_PI**, and **M_2_SQRTPI**).

M_SQRT2 The positive square root of 2.

M_SQRT1_2
The positive square root of 1/2.

For the definitions of various machine-dependent constants, see the description of the **<values.h>** header file.

FILES

```
/usr/include/math.h
```

SEE ALSO

intro(3), fenv(5), values(5).

STANDARDS CONFORMANCE

<math.h>: SVID3, XPG4.2, ANSI C

NAME

mknod.h - header file of macros for handling device numbers

SYNOPSIS

```
#include <sys/mknod.h>
```

DESCRIPTION

The header file `<sys/mknod.h>` defines macros to create and interpret device identification numbers for use with the `mknod()` system call (see *mknod(2)*).

The use of these macros is architecture-dependent. See the System Administration Manual for your system for information on how to select major and minor device numbers.

`mknod.h` contains the macro

```
dev_t makedev(int major, int minor)
```

which packs the major and minor components into a device identification number suitable for the `dev` argument of `mknod()`, and the two macros:

```
int major(dev_t dev)
int minor(dev_t dev)
```

which extract the major and minor number components, respectively, from a device identification number, `dev`.

The macro `MINOR_FORMAT` is a `printf()` specification (see *printf(3S)*) that prints the minor number in the format best suited to the particular implementation; it is used by the long format of the `ls` command (see *ls(1)*) to show the minor numbers for device files.

The base of the number is indicated in the same way as in the C programming language: no leading zero for decimal, leading zero for octal, and leading `0x` for hexadecimal.

SEE ALSO

`ls(1)`, `mknod(1M)`, `mknod(2)`, `printf(3S)`.


m

NAME

mm - the MM macro package for formatting documents

SYNOPSIS

mm [*options*] [*files*]
nroff -**mm** [*options*] [*files*]

DESCRIPTION

This package provides a formatting capability for a very wide variety of documents. The manner in which a document is typed in and edited is essentially independent of whether the document is to be eventually formatted at a terminal or is to be phototypeset. See the references below for further details. The **-mm** option causes *nroff*(1) and *troff* to use the non-compacted version of the macro package.

FILES

/usr/share/lib/macros/mmn	non-compacted version of the package
/usr/share/lib/tmac/tmac.m	pointer to the non-compacted version of the package

SEE ALSO

mm(1), nroff(1).

MM - Memorandum Macros tutorial in *Text Formatters User's Guide*.

NAME

mman - memory mapping definitions

SYNOPSIS

```
#include <sys/mman.h>
```

DESCRIPTION

The `<sys/mman.h>` header defines the following symbolic constants for use with the `madvise()` function:

<code>MADV_NORMAL</code>	No further special treatment.
<code>MADV_RANDOM</code>	Expect random page references.
<code>MADV_SEQUENTIAL</code>	Expect sequential page references.
<code>MADV_WILLNEED</code>	Will need these pages.
<code>MADV_DONTNEED</code>	Will not need these pages.
<code>MADV_SPACEAVAIL</code>	Ensure that resources are reserved.

The following symbolic constants are defined for use with the `mmap()` and `mprotect()` functions:

<code>PROT_READ</code>	Region can be read.
<code>PROT_WRITE</code>	Region can be written.
<code>PROT_EXEC</code>	Region can be executed.
<code>PROT_NONE</code>	Region cannot be accessed.

The following symbolic constants are defined for use with the `mmap()` function:

<code>MAP_FILE</code>	Map a file.
<code>MAP_ANONYMOUS</code>	Map an unnamed memory region.
<code>MAP_VARIABLE</code>	Place region at implementation-computed address.
<code>MAP_FIXED</code>	Place region at specified address.
<code>MAP_SHARED</code>	Share changes made to mapped region.
<code>MAP_PRIVATE</code>	Changes to mapped region are private to a process.

The following symbolic constants are defined for use with the `msync()` function:

<code>MS_SYNC</code>	Perform synchronous writes.
<code>MS_ASYNC</code>	Perform asynchronous writes.
<code>MS_INVALIDATE</code>	Invalidate cached pages.

The following symbolic constants are defined for use with the `msem_init()`, `msem_lock()`, and `msem_unlock()` functions:

<code>MSEM_LOCKED</code>	Create semaphore in locked state.
<code>MSEM_UNLOCKED</code>	Create semaphore in unlocked state.
<code>MSEM_IF_NOWAIT</code>	Do not wait if semaphore is locked.
<code>MSEM_IF_WAITERS</code>	Do not unlock if semaphore has no waiters.

The `typedef struct msemaphore` is defined for use with the `msem_init()`, `msem_lock()`, `msem_unlock()`, and `msem_remove()` functions.

SEE ALSO

`mmap(2)`, `munmap(2)`, `mprotect(2)`, `msync(2)`, `madvise(2)`, `msem_init(2)`, `msem_remove(2)`, `msem_lock(2)`, `msem_unlock(2)`.

NAME

ndir.h - format of HP-UX directory streams

SYNOPSIS

```
#include <ndir.h>
```

DESCRIPTION

This header file defines data types used by the directory stream routines described in *directory(3C)*. It is provided to allow older HP-UX programs to compile unmodified. This file is obsoleted starting from HP-UX 10.30 and is going to be removed in future releases. The header file **<dirent.h>** described on *dirent(5)* should be used in all new programs for compatibility with System V Release 3, the *X/Open Portability Guide*, and the IEEE P1003.1 POSIX standard.

The following data types are defined:

DIR A structure containing information about an open directory stream.

struct direct A structure defining the format of entries returned by the old HP-UX *readdir* function (see *directory(3C)*).

The **struct direct** structure includes the following members:

```
char d_name[MAXNAMLEN+1]; /* name of directory entry */
long d_ino;                /* file serial number */
short d_namlen;            /* length of string in d_name */
short d_reclen;            /* length of this record */
```

The constant **MAXNAMLEN** is defined in **<ndir.h>**.

This file also contains external declarations for the functions in the *directory(3C)* package, including the following declaration:

```
extern struct direct *readdir();
```

WARNINGS

lint(1) might complain about programs that include this file, although they compile and run correctly.

AUTHOR

ndir.h was developed by the University of California, Berkeley, and HP.

SEE ALSO

directory(3C), *dirent(5)*.

NAME

nlio - Native Language I/O (NLIO) Subsystem

DESCRIPTION

The HP-UX Native Language I/O (NLIO) Subsystem is a set of servers, filters, utilities and libraries that provide means to efficiently input and output multibyte characters on multibyte terminals, printers and the X Window System.

NLIO provides application-transparent multibyte code conversion between the internal code and the external code. Application programs including HP-UX commands can utilize the feature of the multibyte character I/O over multibyte terminals, printers and the X Window System without modifying the I/O portion of the program code or linking any special I/O library.

The supported languages are Japanese, Korean, Simplified Chinese and Traditional Chinese.

NLIO supports the following functionalities:

- Read and write multibyte characters on HP Asian terminals and the X Window System.
- Print multibyte characters with HP Asian or HP LaserJet printers.
- Allow HP-UX commands, such as *cat*(1), *more*(1), and *vi*(1), to use multibyte characters.
- Create and modify user-defined characters.
- Support multibyte characters for the *Starbase Graphics Library*.
- Provide language-specific popular input methods.
- Support code conversion utilities and libraries.
- Create and modify user-defined dictionaries for Japanese and Korean.
- Read and write Japanese multibyte characters on a bitmap display.
- Write Japanese multibyte characters on a bitmap display with the FAFM libraries.
- Provide a set of libraries for Japanese Kana-to-Kanji conversion.

AUTHOR

nlio was developed by HP.

SEE ALSO

Native Language I/O manuals

n

NAME

pam_unix - authentication, account, session, and password management PAM modules for UNIX

SYNOPSIS

/usr/lib/security/libpam_unix.1

DESCRIPTION

The UNIX service module for PAM, /usr/lib/security/libpam_unix.1, provides functionality for all four PAM modules: authentication, account management, session management and password management.

The libpam_unix.1 module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

Unix Authentication Module

The UNIX authentication component provides functions to verify the identity of a user, (pam_sm_authenticate()) and to set user specific credentials (pam_sm_setcred()).

pam_sm_authenticate() compares the user entered password (or password retrieved from the user's smart card) with the password from UNIX password database, including the protected password database for trusted systems. If the passwords match, the user is authenticated. If the user also has secure RPC credentials and the secure RPC password is the same as the UNIX password, then the secure RPC credentials are also obtained.

The following options may be passed to the UNIX service module:

debug *syslog*(3C) debugging information at LOG_DEBUG level.

nowarn Turn off warning messages.

use_first_pass

It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, quit and do not prompt the user for a password. This option should only be used if the authentication service is designated as *optional* in the **pam.conf** configuration file.

try_first_pass

It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, prompt the user for a password.

use_psd

psd stands for personal security device, for the current implementation there is only one security device: the smart card. It compares the password in the password database with the password stored on the user's smart card. With this option the PAM Framework prompt "Enter PIN:" is used instead of the password prompt. This option is only supported with the authentication or password module types (auth, password) services in the **pam.conf** or in the **pam_user.conf** configuration files.

When prompting for the current password, the UNIX authentication module will use the prompt, "Password:" unless one of the following scenarios occur:

1. The option **try_first_pass** is specified and the password entered for the first module in the stack fails for the UNIX module.
2. The option **try_first_pass** is not specified, and the earlier authentication modules listed in the **pam.conf** file have prompted the user for the password.
3. The option **use_psd** is specified. In this case, the UNIX authentication module will use the prompt "Enter PIN:".

In cases 1 and 2, the UNIX authentication module will use the prompt "System Password:".

The **pam_sm_setcred()** function sets user specific credentials. If the user had secure RPC credentials, but the secure RPC password was not the same as the UNIX password, then a warning message is printed. If the user wants to get secure RPC credentials, then *keylogin*(1) needs to be run.

Unix Account Management Module

The UNIX account management component provides a function to perform account management (`pam_sm_acct_mgmt()`). The function retrieves the user's password entry from the UNIX password database and verifies that the user's account and password have not expired. For trusted systems, this module also validates the allowed access time and access terminal based upon the security configuration. The following options may be passed in to the UNIX service module:

debug *syslog*(3C) debugging information at LOG_DEBUG level.
nowarn Turn off warning messages.

Unix Session Management Module

The UNIX session management component provides functions to initiate (`pam_sm_open_session()`) and terminate (`pam_sm_close_session()`) UNIX sessions. For UNIX, `pam_open_session()` updates the last successful or unsuccessful login time in the protected password database for trusted mode. The account management module reads the information to display the previous time the user logged in. The following options may be passed in to the UNIX service module:

debug *syslog*(3C) debugging information at LOG_DEBUG level.
nowarn Turn off warning messages.
pam_close_session is a NULL function.

Unix Password Management Module

The UNIX password management component provides a function to change passwords (`pam_sm_chauthtok()`) in the UNIX password database. This module must be **required** in `pam.conf`. It can not be **optional** or **sufficient**. The following options may be passed in to the UNIX service module:

debug *syslog*(3C) debugging information at LOG_DEBUG level.
nowarn Turn off warning messages.
use_first_pass

It compares the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, quit and do not prompt the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, quit and do not prompt the user for a new password.

try_first_pass

It compares the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, prompt the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, prompt the user for a new password.

use_psd

It prompts the user for the PIN (with the PIN, the PAM Framework can retrieve a password from the smart card) and the old password is retrieved from the smart card. It compares the password in the password database with the user's old password. If the passwords match, it prompts the user for a new password.

If the user's password has expired, the UNIX account module saves this information in the authentication handle using `pam_set_data()`. The UNIX password module retrieves this information from the authentication handle using `pam_get_data()` to determine whether or not to force the user to update their password.

SEE ALSO

keylogin(1), pam(3), pam_authenticate(3), pam_setcred(3), syslog(3C), pam.conf(4), pam_user.conf(4).

NAME

pam_updbe - user policy definition service module

SYNOPSIS

`/usr/lib/security/libpam_updbe.1`

DESCRIPTION

The user policy definition service module for PAM, `/usr/lib/security/libpam_updbe.1`, reads *options* defined in the user configuration file, `/etc/pam_user.conf` (see *pam_user.conf*(4)) and uses `pam_set_data` (see *pam_set_data*(3)) to store the information in the *pam handle* for subsequent service modules to use. Service modules perform `pam_get_data` to retrieve corresponding information from the *pam handle* when the user is authenticated, or when the user password is changed.

The use of `pam_updbe` is not mandatory. It is needed only when per user configuration is used. However, in order for the functionality of `pam_updbe` to take effect, it must be listed as the first service module in `/etc/pam.conf`.

Like any other service module, `pam_updbe` provides interfaces for all four PAM modules: authentication, account management, session management and password management. Each module just reads the options defined for the specific module type.

UPDBE Authentication Module

The UPDBE authentication component provides functions to read options defined in `pam_user.conf` for the module type "auth". The module data name used is `PAM_AUTH_USER`.

Unix Account Management Module

The UNIX account management component provides a function to read options defined in `pam_user.conf` for the module type "account". The module data name used is `PAM_ACCOUNT_USER`.

Unix Session Management Module

The UNIX session management component provides a function to read options defined in `pam_user.conf` for the module type "session". The module data name used is `PAM_SESSION_USER`.

Unix Password Management Module

The UNIX password management component provides a function to read options defined in `pam_user.conf` for the module type "password". The module data name used is `PAM_PASSWORD_USER`.

SEE ALSO

`pam`(3), `pam_set_data`(3), `pam.conf`(4), `pam_user.conf`(4).

NAME

pd_att - index of HPDPS attribute manpages

DESCRIPTION

To view the file that contains information about the valid attributes for a given object, enter the following command:

man *name_of_file*

where *name_of_file* is one of the following:

pd_att_document
pd_att_ivdocument
pd_att_ivjob
pd_att_job
pd_att_log
pd_att_queue
pd_att_log_ptr
pd_att_phy_ptr
pd_att_spooler
pd_att_supervisor

NAME

pd_att_document - list of attributes for an HPDPS document object

DESCRIPTION

The following is a list of valid attributes and values for the document object class of the HP Distributed Print Service.

Attributes for a Document Object

Settable Attribute Listing	Specifiable Attribute Listing
bottom-margin content-orientation copy-count default-character-set default-font default-input-tray default-medium default-printer-resolution document-comment document-format input-tray-select left-margin number-up page-count page-select plex printer-pass-through right-margin sides top-margin	document-file-name document-type initial-value-document transfer-method

Table 1-1. Document Attributes

Attribute	Description
Name	beginning-page-requested
Type	Non Settable.
Usage	
Explanation	Identifies the beginning page of the document specified by the beginning-page component of the document page-select attribute.
Value Type	Single Value.
Values	An integer from 1 to 2147483647. Implicitly set by HPDPS.
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> This attribute is intended for use in the ipmap file to convert the beginning-page component of the page-select attribute to the interface program option. Use the page-select attribute to specify the starting page to print with the pdpr command.

Attribute	Description
Name	bottom-margin
Input Synonym	bm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the bottom edge of the logical page and the bottom edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	content-orientation
Input Synonym	orientation
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies the page presentation (how the data is to be placed on a page) for the document.
Value Type	Single value.
Values	One of the following fixed values: landscape portrait reverse-portrait reverse-landscape
Default	portrait
Usage Guidelines	o The document is validated and scheduled against the printer attribute content-orientations-supported.
Attribute	Description
Name	copy-count
Input Synonym	copies

Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the number of document copies to be printed per job copy.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	1
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated and scheduled against the printer attribute maximum-copies-supported. o A value of zero (0) is an error.
Attribute	Description
Name	default-character-set
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies the default character set to be used.
Value Type	Single value.
Values	<p>One of the following fixed values:</p> <pre> iso-ucs-2-level2 other ascii iso-latin1 iso-latin2 iso-latin3 iso-latin4 iso-latin-cyrillic iso-latin-arabic iso-latin-greek iso-latin-hebrew iso-latin5 iso-latin6 iso-text-communication half-width-katakana jis-encoding shift-jis euc-packed-format-japanese euc-fixed-width-japanese iso-reg4-united-kingdom iso-reg11-swedish-for-names iso-reg15-italian iso-reg17-spanish iso-reg21-german iso-reg60-danish-norwegian iso-reg69-french unicode ucs4 unicode-ascii unicode-latin1 </pre>

	unicode-ibm-2039 unicode-ibm-1261 unicode-ibm-1268 unicode-ibm-1276 unicode-ibm-1264 unicode-ibm-1265 windows30-latin1 windows31-latin1 windows31-latin2 windows31-latin5 hp-roman8 adobe-standard-encoding ventura-us ventura-international dec-mcs pc-page-850-multilingual pc-page-852-latin2 pc8-page-437-us pc8-danish-norwegian pc-page-862-latin-hebrew pc8-turkish ibm-symbols ibm-thai hp-legal hp-pifont hp-math8 hp-ps-math hp-desktop ventura-math microsoft-publishing windows31j gb-2312 big5 cns ksc-5601 cccl
Default	No default.
Usage Guidelines	o The document is validated and scheduled against the printer attribute character-sets-supported.
Attribute	Description
Name	default-font
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies the default font to be used.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the font name.
Default	No default.
Usage Guidelines	o The document is validated and scheduled against the printer attribute fonts-supported.

Attribute	Description
Name	default-input-tray
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies an input-tray on the printer that contains the medium that is to be used for normal document pages.
Value Type	Single value.
Values	<p>A text string of up to 255 characters that describes the input tray, or one of the following fixed values:</p> <p>top middle bottom envelope manual large-capacity main side</p>
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated and scheduled against the printer attribute input-trays-supported. o The following attributes may be used to select the input tray. If more than one of these attributes are supplied then the attribute with the highest precedence, as numbered, is chosen: <ul style="list-style-type: none"> 1. input-tray-select 2. default-medium 3. default-input-tray
Attribute	Description
Name	default-medium
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies the medium for normal document pages on which this document is to be printed.
Value Type	Single value.
Values	<p>One of the following fixed values or a text string that contains the name of the medium. The text string can be up to 255 characters long per medium.</p> <p>iso-a4-white iso-a4-colored iso-a4-transparent iso-a3-white</p>

iso-a3-colored
iso-a5-white
iso-a5-colored
iso-b4-white
iso-b4-colored
iso-b5-white
iso-b5-colored
na-letter-white
na-letter-colored
na-letter-transparent
na-legal-white
na-legal-colored
iso-b5-envelope
iso-b4-envelope
iso-c4-envelope
iso-c5-envelope
designated-long-envelope
na-10x13-envelope
na-9x12-envelope
monarch-envelope
na-number-10-envelope
na-7x9-envelope
na-9x11-envelope
na-10x14-envelope
na-number-9-envelope
na-6x9-envelope
na-10x15-envelope
jis-b4-white
jis-b4-colored
jis-b5-white
jis-b5-colored
executive
folio
invoice
ledger
quarto
iso-a0-white
iso-a1-white
iso-a2-white
iso-a6-white
iso-a7-white
iso-a8-white
iso-a9-white
iso-a10-white
iso-b10-white
iso-b0-white
iso-b1-white
iso-b2-white
iso-b3-white
iso-b6-white
iso-b7-white
iso-b8-white
iso-b9-white
iso-b10-white
jis-b0-white
jis-b1-white
jis-b2-white
jis-b3-white
jis-b6-white
jis-b7-white
jis-b8-white
jis-b9-white



p

	jis-b10-white a b c d e
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated against the logical printer attribute media-supported and the physical printer attribute media-supported o The document is scheduled against the physical printer attributes media-supported and media-ready o The following attributes may be used to select the input tray. If more than one of these attributes are supplied then the attribute with the highest precedence, as numbered, is chosen: <ul style="list-style-type: none"> 1. input-tray-select 2. default-medium 3. default-input-tray
Attribute	Description
Name	default-printer-resolution
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies the default printer resolution to be used.
Value Type	Single Value.
Values	<p>One of the following fixed values or an integer from 1 to 2147483647.</p> <p>highest lowest middle</p>
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The value of this attribute is validated against the printer attribute printer-resolutions-supported.
Attribute	Description
Name	document-comment
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Provides information that is associated with this document.
Value Type	Single value.

Values	A text string up to 4095 characters that contains information about this document such as the fonts to be used.																		
Default	No default.																		
Usage Guidelines																			
Attribute	Description																		
Name	document-file-name																		
Input Synonym	file-name																		
Type	Specifiable																		
Usage																			
Explanation	Specifies the file name or other source that provided the data for the document to be or was printed.																		
Value Type	Single value.																		
Values	A text string up to 4095 characters that contains the global ID.																		
Default	No default.																		
Usage Guidelines	<ul style="list-style-type: none"> Set by HPDPS from the name of the file or source specified with the pdpr command. 																		
Attribute	Description																		
Name	document-format																		
Input Synonym	format																		
Type	Settable with the pdpr, pdset, or pdmod commands.																		
Usage	Values vary by printer model.																		
Explanation	Identifies the format (datatype) of the document.																		
Value Type	Single value.																		
Values	<p>One of the following fixed values:</p> <table> <tr> <th>Fixed Value</th><th>Input Synonym</th></tr> <tr> <td>ascii</td><td></td></tr> <tr> <td>pcl</td><td>hppcl</td></tr> <tr> <td></td><td>hp-pcl</td></tr> <tr> <td>hpgl</td><td>hp-gl</td></tr> <tr> <td>pjl</td><td></td></tr> <tr> <td>postscript</td><td>ps</td></tr> <tr> <td>ipds</td><td></td></tr> <tr> <td>ppds</td><td></td></tr> </table>	Fixed Value	Input Synonym	ascii		pcl	hppcl		hp-pcl	hpgl	hp-gl	pjl		postscript	ps	ipds		ppds	
Fixed Value	Input Synonym																		
ascii																			
pcl	hppcl																		
	hp-pcl																		
hpgl	hp-gl																		
pjl																			
postscript	ps																		
ipds																			
ppds																			

```

escapep
epson
ddif
interpress
iso-6429
line-data
modca
regis
scs
spdl
tek4014
pds
igp
codev
dsc-dse
wps
ln03
ccitt
quic
cpap
dec-ppl
simple-text
npap
doc
impress
pinwriter
nmdl
nec201pl
automatic
pages
lips
tiff
diagnostic
capsl
excl
lcds
xes
passthru

```

Default	ascii
---------	-------

Usage Guidelines	o The document is validated and scheduled against the printer attribute document-formats-supported.
------------------	---

Attribute	Description
-----------	-------------

Name	document-sequence-number
------	--------------------------

Input Synonym	sequence-number, document-number
------------------	----------------------------------

Type	Not Settable
------	--------------

Usage	
-------	--

Explanation	Specifies the document number in relation to the other documents of a multiple document job. HPDPS sets this value.
-------------	---

Value Type	Single value.
------------	---------------

Values	One of the integers from 1 to 2147483647.
Default	No default.
Usage Guidelines	o Used to identify this document in request status or when HPDPS provides event status.
Attribute	Description
Name	document-type
Input Synonym	type
Type	Specifiable using the pdpr command.
Usage	Values vary by printer model.
Explanation	Indicates that the document is either a printable document, a font, or some other resource.
Value Type	Single value.
Values	One of the following fixed values: file-reference printable
Default	printable
Usage Guidelines	o The document is validated and scheduled against the printer attribute document-types-supported.
Attribute	Description
Name	ending-page-requested
Type	Non Settable.
Usage	
Explanation	Identifies the ending page of the document specified by the ending-page component of the document page-select attribute.
Value Type	Single Value.
Values	An integer from 1 to 2147483647. Implicitly set by HPDPS.
Default	No default
Usage Guidelines	o This attribute is intended for use in the ipmap file to convert the ending-page component of the page-select attribute to the interface program option. Use the page-select attribute to specify the ending page to print with the pdpr command.
Attribute	Description

Name	initial-value-document
Type	Specifiable with the pdpr command.
Usage	
Explanation	Identifies an initial-value-document object (within a given spooler) that is to be used for this document. This object may be used when a document is created during job submission.
Value Type	Single value.
Values	Any text string up to 255 characters that contains the name of this initial-value-document.
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o If this attribute is specified, the document uses the attribute values from the initial-value-document object to set the document attribute values, unless initial-value-document attribute values are overridden by attribute values supplied at the command line.
Attribute	Description
Name	input-tray-select
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Identifies which input tray (bin) contains the medium that is to be used for this document.
Value Type	Single value.
Values	<p>A text string of up to 255 characters that describes the input tray, or one of the following fixed values:</p> <p>top middle bottom envelope manual large-capacity main side</p>
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated and scheduled against the printer attribute input-trays-supported. o The following attributes may be used to select the input tray. If more than one of these attributes are supplied then the attribute with the highest precedence, as numbered, is chosen:

1. input-tray-select

	2. default-medium 3. default-input-tray
Attribute	Description
Name	left-margin
Input Synonym	lm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the left edge of the logical page and the left edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	number-up
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Specifies the number of source page-images to impose upon a single instance of a selected medium, that is, the number of source pages to appear on a single printed page. The possible values are: 0 Suppress any number-up operation or embellishment. 1 1 source page per physical page. 2 2 source pages per physical page. 4 4 source pages per physical page.
Value Type	Single value.
Values	One of the following fixed values: 0 1 2 4
Default	No default.
Usage Guidelines	

Attribute	Description
Name	object-class
Type	Non Settable
Usage	
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is document.
Default	document
Usage Guidelines	
Attribute	Description
Name	octet-count
Type	Non Settable
Usage	
Explanation	Specifies the document size in octets (bytes). Computed by HPDPS when this document object is created.
Value Type	Single value.
Values	An integer from 0 to 9223372036854775800.
Default	N/A
Usage Guidelines	o Used to compute total octet count for a job and to provide information about this document.
Attribute	Description
Name	page-count
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Specifies the estimated length of the job in pages.
Explanation	HPDPS determines the size of a job based on the total number of octets (bytes) in the job. For operator job management purposes, you may find that job size based on pages is easier.
Value Type	Single value.
Values	An integer from 0 to 2147483647.
Default	No default.

Usage Guidelines	Because HPDPS does not estimate job size in pages or use the page-count value you specify, the value for the page-count attribute should closely represent the actual number of pages in the job if the operator is to make valid decisions based on page count.
Attribute	Description
Name	page-select
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Support for this attribute varies by printer model. The value syntax also varies by printer model.
Explanation	Specifies the range of pages to be printed.
Value Type	Multiple values, however, only the first value is currently used.
Values	This is a complex attribute which has the following components: <ol style="list-style-type: none"> 1. beginning-page (an integer page number) 2. ending-page (an integer page number)
Syntax	-x "page-select=beginning-page:ending-page" <p>For example: -x "page-select=10:30" prints only pages 10 through 30.</p>
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated against the printer attribute page-select-supported. o Only a numeric value is supported for this attribute. o Only one page range may be specified.
Component	Description
beginning-page	An integer from 1 to 2147483647.
ending-page	An integer from 1 to 2147483647.
Attribute	Description
Name	plex
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Indicates whether the page images of this document are conditioned for one-sided or two-sided printing and the relative orientation of consecutive pages.
Value Type	Single value.

Values	One of the following fixed values: simplex (single sided) duplex (double sided) tumble (double sided, every other page upside-down)
Default	simplex
Usage Guidelines	<ul style="list-style-type: none"> o This document is validated and scheduled against the printer attribute plexes-supported. o HPDPS defaults "sides=2" when plex=duplex. HPDPS does not currently perform any other conditioning when plex=duplex.
Attribute	Description
Name	printer-initial-value-document
Type	Non Settable
Usage	
Explanation	HPDPS sets this value to the name of the initial-value-document object used.
Value Type	Single value.
Default	No default.
Attribute	Description
Name	printer-pass-through
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Allows specific printer information to be submitted along with the document.
Value Type	Single value.
Values	A text string up to 4095 characters that contains options passed directly to the interface program.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The value of this attribute is passed as the 4th argument of the printer interface program.
Attribute	Description
Name	right-margin
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the right edge of the

	logical page and the right edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	sides
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the number of media sides on which the document is to be printed.
Value Type	Single value.
Values	An integer of 1 or 2.
Default	1
Usage Guidelines	o This document is validated and scheduled against the printer attribute sides-supported.
Attribute	Description
Name	top-margin
Input Synonym	tm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the bottom edge of the logical page and the bottom edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description

Name	transfer-method
Type	Specifiable with the pdpr command.
Usage	
Explanation	Identifies the method by which the document is transferred to, or acquired by, the print server.
Value Type	Single value.
Values	One of the following fixed values: dce-pipe-pull with-request
Default	dce-pipe-pull
Usage Guidelines	o This document is validated and scheduled against the supervisor attribute transfer-methods-supported.

p

NAME

pd_att_ivdocument - list of attributes for an HPDPS initial value document object

DESCRIPTION

The following is a list of valid attributes and values for the initial value document object class of the HP Distributed Print Service.

Initial-value-document objects are used to set default values for document attributes.

The *initial-value-document* object contains two types of attributes:

- 1. Those that either describe or are used by the *initial-value-document* object itself.
- 2. Those used for defaulting values for document attributes.

Attributes for an Initial Value Document Object

Settable Attribute Listing	Specifiable Attribute Listing
bottom-margin content-orientation copy-count default-character-set default-font default-input-tray default-medium default-printer-resolution descriptor document-comment document-format document-type input-tray-select left-margin list-of-managers number-up message page-select plex printer-pass-through right-margin sides top-margin	There are no specifiable attributes for a initial-value-document object.

Note: In the following table, the attributes listed are used to set or used as defaults for the document unless otherwise specified.

Table 1-1. Initial Value Document Attributes	
Attribute	Description
Name	associated-server
Type	Non Settable
Usage	
Explanation	Indicates the name of the spooler in which this initial-value-document object resides.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name



	of the server. Set by HPDPS to the ServerName: portion of the argument when this initial-value-document was created.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> This attribute is used by the initial-value-document object. Value was set to the ServerName specified in the pdcreate command when this initial-value object was created.
Attribute	Description
Name	bottom-margin
Input Synonym	bm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the bottom edge of the logical page and the bottom edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	content-orientation
Input	orientation
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Identifies the page presentation (how the data is to be placed on a page) for the document.
Value Type	Single value.
Values	One of the following fixed values: landscape portrait reverse-portrait reverse-landscape
Default	No default.

Usage Guidelines	o The document is validated and scheduled against the printer attribute content-orientations-supported.
Attribute	Description
Name	copy-count
Input Synonym	copies
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Specifies the number of document copies to be printed per job copy.
Value Type	Single value.
Values	An integer from 1 to 2147483647
Default	No default.
Usage Guidelines	o A value of zero (0) is an error. o The document is validated and scheduled against the printer attribute maximum-copies-supported.
Attribute	Description
Name	default-character-set
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model
Explanation	Identifies a coded character-set that the server shall use as the coded character-set default for the pages of the document that require a coded character set specification.
Value Type	Single value.
Values	One of the following fixed values: iso-ucs-2-level2 other ascii iso-latin1 iso-latin2 iso-latin3 iso-latin4 iso-latin-cyrillic iso-latin-arabic iso-latin-greek iso-latin-hebrew iso-latin5 iso-latin6 iso-text-communication half-width-katakana jis-encoding

```

shift-jis
euc-packed-format-japanese
euc-fixed-width-japanese
iso-reg4-united-kingdom
iso-reg11-swedish-for-names
iso-reg15-italian
iso-reg17-spanish
iso-reg21-german
iso-reg60-danish-norwegian
iso-reg69-french
unicode
ucs4
unicode-ascii
unicode-latin1
unicode-ibm-2039
unicode-ibm-1261
unicode-ibm-1268
unicode-ibm-1276
unicode-ibm-1264
unicode-ibm-1265
windows30-latin1
windows31-latin1
windows31-latin2
windows31-latin5
hp-roman8
adobe-standard-encoding
ventura-us
ventura-international
dec-mcs
pc-page-850-multilingual
pc-page-852-latin2
pc8-page-437-us
pc8-danish-norwegian
pc-page-862-latin-hebrew
pc8-turkish
ibm-symbols
ibm-thai
hp-legal
hp-pifont
hp-math8
hp-ps-math
hp-desktop
ventura-math
microsoft-publishing
windows31j
gb-2312
big5
cns
ksc-5601
ccdc

```

Default

No default.

Usage
Guidelines

o The document is validated and scheduled against the printer attribute character-sets-supported.

Attribute

Description

Name

default-font

Type

Settable with the pdpr, pdset, or pdmod commands.

Usage	Values vary by printer model.
Explanation	Identifies the default font to be used.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the font name.
Default	No default.
Usage Guidelines	o The document is validated and scheduled against the printer attribute fonts-supported.
Attribute	Description
Name	default-input-tray
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Identifies an input-tray that contains the medium to be used for the document.
Value Type	Single value.
Values	One of the following fixed values: top middle bottom envelope manual large-capacity main side
Default	No default.
Usage Guidelines	o The document is validated and scheduled against the printer attribute input-trays-supported.
Attribute	Description
Name	default-medium
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Identifies the medium to be used for this document.
Value Type	Single value.
Values	One of the following fixed values or a text string that contains the name of the medium. The text string can be up to 255 characters long per medium.

iso-a4-white
 iso-a4-colored
 iso-a4-transparent
 iso-a3-white
 iso-a3-colored
 iso-a5-white
 iso-a5-colored
 iso-b4-white
 iso-b4-colored
 iso-b5-white
 iso-b5-colored
 na-letter-white
 na-letter-colored
 na-letter-transparent
 na-legal-white
 na-legal-colored
 iso-b5-envelope
 iso-b4-envelope
 iso-c4-envelope
 iso-c5-envelope
 designated-long-envelope
 na-10x13-envelope
 na-9x12-envelope
 monarch-envelope
 na-number-10-envelope
 na-7x9-envelope
 na-9x11-envelope
 na-10x14-envelope
 na-number-9-envelope
 na-6x9-envelope
 na-10x15-envelope
 jis-b4-white
 jis-b4-colored
 jis-b5-white
 jis-b5-colored
 executive
 folio
 invoice
 ledger
 quarto
 iso-a0-white
 iso-a1-white
 iso-a2-white
 iso-a6-white
 iso-a7-white
 iso-a8-white
 iso-a9-white
 iso-a10-white
 iso-b0-white
 iso-b1-white
 iso-b2-white
 iso-b3-white
 iso-b6-white
 iso-b7-white
 iso-b8-white
 iso-b9-white
 iso-b10-white
 jis-b0-white
 jis-b1-white
 jis-b2-white
 jis-b3-white

	jis-b6-white jis-b7-white jis-b8-white jis-b9-white jis-b10-white a b c d e
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated against the logical printer attribute media-supported and the physical printer attribute media-supported. o The document is scheduled against the physical printer attributes media-supported and media-ready.
Attribute	Description
Name	default-printer-resolution
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Identifies the default printer resolution to be used.
Value Type	Single Value.
Values	One of the following fixed values or an integer from 1 to 2147483647. highest lowest middle
Default	No default.
Usage Guidelines	
Attribute	Description
Name	descriptor
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Provides a description of this initial-value-document.
Value Type	Single value.
Values	A text string up to 4095 characters that describes this initial value document.
Default	No default.

Usage Guidelines	o This attribute is used by the initial-value-document object.																																
Attribute	Description																																
Name	document-comment																																
Type	Settable with the pdcreate or pdset command.																																
Usage																																	
Explanation	Provides information that is associated with this initial-value-document.																																
Value Type	Single value.																																
Values	A text string up to 4095 characters that contains information that is associated with this initial value document.																																
Default	No default.																																
Usage Guidelines																																	
Attribute	Description																																
Name	document-format																																
Input Synonym	format																																
Type	Settable with the pdcreate or pdset command.																																
Usage	Values vary by printer model.																																
Explanation	Identifies the format of this document.																																
Value Type	Single value.																																
Values	One of the following fixed values:																																
	<table> <tr> <th>Fixed Value</th><th>Input Synonym</th></tr> <tr> <td>ascii</td><td></td></tr> <tr> <td>pcl</td><td>hppcl hp-pcl</td></tr> <tr> <td>hpgl</td><td>hp-gl</td></tr> <tr> <td>pjl</td><td></td></tr> <tr> <td>postscript</td><td>ps</td></tr> <tr> <td>ipds</td><td></td></tr> <tr> <td>ppds</td><td></td></tr> <tr> <td>escapep</td><td></td></tr> <tr> <td>epson</td><td></td></tr> <tr> <td>ddif</td><td></td></tr> <tr> <td>interpress</td><td></td></tr> <tr> <td>iso-6429</td><td></td></tr> <tr> <td>line-data</td><td></td></tr> <tr> <td>modca</td><td></td></tr> <tr> <td>regis</td><td></td></tr> </table>	Fixed Value	Input Synonym	ascii		pcl	hppcl hp-pcl	hpgl	hp-gl	pjl		postscript	ps	ipds		ppds		escapep		epson		ddif		interpress		iso-6429		line-data		modca		regis	
Fixed Value	Input Synonym																																
ascii																																	
pcl	hppcl hp-pcl																																
hpgl	hp-gl																																
pjl																																	
postscript	ps																																
ipds																																	
ppds																																	
escapep																																	
epson																																	
ddif																																	
interpress																																	
iso-6429																																	
line-data																																	
modca																																	
regis																																	

	scs spd1 tek4014 pds igp codev dsc-dse wps ln03 ccitt quic cpap dec-ppl simple-text npap doc impress pinwriter npd1 nec201pl automatic pages lips tiff diagnostic caps1 excl lcds xes passthru
Default	ascii
Usage Guidelines	o The document is validated and scheduled against the printer attribute document-formats-supported.
Attribute	Description
Name	document-type
Input Synonym	type
Type	Settable with the pdcreate or pdset command.
Usage	Values vary by printer model.
Explanation	Indicates that the document is either a printable document, a font, or some other resource.
Value Type	Single value.
Values	One of the following fixed values: file-reference printable
Default	No default.
Usage Guidelines	o The document is validated and scheduled against the printer attribute document-types-supported.

Attribute	Description
Name	initial-value-document-identifier
Type	Specifiable with the pdcreate command.
Usage	
Explanation	Identifies this initial-value-document object.
Value Type	Single value.
Values	A text string that contains the identification of this initial-value-document. Specified in the argument ServerName:InitialValueDocumentName of the pdcreate command.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-document object. o The value for this attribute must be unique within the spooler.
Attribute	Description
Name	input-tray-select
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Identifies which input tray contains the medium to be used for this document.
Value Type	Single value.
Values	One of the following fixed values: top middle bottom envelope manual large-capacity main side
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This document is validated and scheduled against the printer attribute input-trays-supported.
Attribute	Description
Name	number-up

Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	Values vary by printer model.
Explanation	Specifies the number of source page-images to impose upon a single instance of a selected medium, that is, the number of source pages to appear on a single printed page. The possible values are: <div> 0 Suppress any number-up operation or embellishment. 1 1 source page per physical page. 2 2 source pages per physical page. 4 4 source pages per physical page. </div>
Value Type	Single value.
Values	One of the following fixed values: <div> 0 1 2 4 </div>
Default	No default.
Usage Guidelines	
Attribute	Description
Name	left-margin
Input Synonym	lm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the left edge of the logical page and the left edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	list-of-managers
Input Synonym	managers

Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Lists the people responsible for the maintenance of this initial-value-document object.
Value Type	Multiple values.
Values	A text string up to 255 characters per person that contains the name or user ID of the person responsible for this initial-value-document.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-document object.
Attribute	Description
Name	logical-printers-ready
Type	Non Settable
Usage	
Explanation	Lists the enabled logical printers that reference this initial-value-document.
Value Type	Multiple values.
Values	A text string up to 255 characters per logical printer that contains the global IDs of the logical printers.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-document object. o Set by HPDPS when the printer attribute printer-initial-value-document of an enabled logical printer references this initial-value-document. o The printer identification is removed from the list if the printer is disabled. o This IVD object cannot be deleted while any of the printers identified in this list are enabled.
Attribute	Description
Name	message
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Provides a message associated with this initial-value-document.
Value Type	Single value.

Values	A text string up to 4095 characters that contains some information about this initial-value-document object.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-document object.
Attribute	Description
Name	object-class
Type	Non Settable
Usage	
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is initial value document.
Default	initial-value-document
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-document object.
Attribute	Description
Name	page-select
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the range of pages to be printed.
Value Type	Multiple values, however, only the first value is currently used.
Values	<p>This is a complex attribute which has the following components:</p> <ol style="list-style-type: none"> 1. beginning-page (an integer page number) 2. ending-page (an integer page number)
Syntax	<p>-x "page-select=beginning-page:ending-page"</p> <p>For example: -x "page-select=10:30" prints only pages 10 through 30.</p>
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document is validated against the printer attribute page-select-supported. o Only numeric value is supported for this attribute. o Only one page range may be specified.
Component	Description

beginning-page	An integer from 1 to 2147483647.
ending-page	An integer from 1 to 2147483647.
Attribute	Description
Name	plex
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Indicates whether the page images of the document are conditioned for one-sided or two-sided printing and the relative orientation of consecutive pages.
Value Type	Single value.
Values	One of the following fixed values: simplex (single sided) duplex (double sided) tumble (double sided, every other page upside-down)
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This document is validated and scheduled against the printer attribute plexes-supported. o HPDPS defaults "sides=2" when plex=duplex. HPDPS does not currently perform any other conditioning when plex=duplex.
Attribute	Description
Name	printer-pass-through
Input Synonym	other-options
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Allows specific printer information to be submitted along with the document.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the DSS information.
Default	No default.
Usage Guidelines	
Attribute	Description

Name	right-margin
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the right edge of the logical page and the right edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	sides
Type	Settable with the pdcreate or pdset command.
Usage	
Explanation	Specifies whether the document is to be printed on one side or both sides of the media.
Value Type	Single value.
Values	An integer of 1 or 2.
Default	1
Usage Guidelines	<ul style="list-style-type: none"> The document is validated and scheduled against the printer attribute sides-supported.
Attribute	Description
Name	top-margin
Input Synonym	tm
Type	Settable with the pdpr, pdset, or pdmod commands.
Usage	
Explanation	Specifies the distance between the top edge of the logical page and the top edge of the text area when held in the reading orientation. The values are in units of characters.
Value Type	Single value.
Values	An integer from 1 to 2147483647.

Default	No default.
Usage	
Guidelines	



p

NAME

pd_att_ivjob - list of attributes for an HPDPS initial value job object

DESCRIPTION

The following is a list of valid attributes and values for the initial value job object class of the HP Distributed Print Service. *Initial-value-job* objects are used to set values for job attributes.

Attributes for an Initial Value Job Object

Settable Attribute Listing	Specifiable Attribute Listing
descriptor interface-program-method job-batch job-comment job-client-id job-discard-time job-end-message job-finishing job-hold job-message-to-operator job-name job-originator job-owner job-print-after job-priority job-retention-period job-sheets job-start-message list-of-managers message notification-profile physical-printers-requested printer-locations-requested printer-models-requested printer-types-requested results-profile	There are no specifiable attributes for a initial-value-document object.

Note: In the following table, the attributes are used to set job attributes or are used as defaults for a job unless otherwise noted.

Table 1-1. Initial Value Job Attributes	
Attribute	Description
Name	associated-server
Type	Non Settable
Explanation	Identifies the name of the spooler in which this initial-value-job object resides.
Value Type	Single value.
Values	A text string that contains the server name. Set to the server name of the pdcreate command argument ServerName:InitialValueJobName when this object is created.

Default	No default.
Usage Guidelines	o This attribute is used by the initial-value-job object.
Attribute	Description
Name	descriptor
Type	Settable with the pdcreate or pdset commands.
Explanation	Provides a description of this initial-value-job.
Value Type	Single value.
Values	A text string up to 4095 characters that describes this initial-value-job object.
Default	No default.
Usage Guidelines	o This attribute is used by the initial-value-job object.
Attribute	Description
Name	initial-value-job-identifier
Type	Non Settable
Explanation	Identifies this initial-value-job object within the spooler.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this initial-value-job object. Set to the name used in the pdcreate command argument ServerName:InitialValueJobName when this object was created.
Default	No default.
Usage Guidelines	o This attribute is used by the initial-value-job object. o The value for this attribute must be unique within a spooler.
Attribute	Description
Name	interface-program-method
Input Synonym	ip-method
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies the invocation method of the interface program.
Value Type	Single value.

Values	One of the following fixed values: hpdps lp
Default	hpdps
Usage Guidelines	<ul style="list-style-type: none"> o The job is validated against the printer attribute interface-program-methods-supported. o If the value hpdps is specified, the interface program is invoked to print one job result-set as follows: <ul style="list-style-type: none"> - The value of the job-copies component of the job results-profile attribute is passed as the 4th argument of the interface program. - Each document is duplicated n times before being passed to the interface program, where n is the value of the document copy-count attribute. o If the value lp is specified, the interface program is invoked to print each job copy in one job result set: <ul style="list-style-type: none"> - The value of document attribute copy-count is passed as the 4th argument of the interface program. - The interface program is invoked n times, where n is the value of the job-copies component of the job results-profile attribute.
Attribute	Description
Name	job-batch
Type	Settable
Explanation	Specifies that the job you are submitting is being marked as a specific type of job. HPDPS processes this job only on a physical printer which is ready to print that specified job-batch type.
Value Type	Single value.
Values	You can enter a value up to 4095 characters that contains the job-batch name.
Default	No default.
Usage Guidelines	HPDPS schedules this attribute against the physical printer's job-batches-ready attribute.
Attribute	Description
Name	job-client-id
Input Synonym	local-id
Type	Settable with the pdcreate or pdset commands.
Explanation	The local job identifier.
Value Type	Single value.

Values	A text string up to 255 characters that contains the local ID (an integer). This value is generated automatically by the DPS client program and is unique for a given user ID. The local ID is mapped to a global ID (the job-identifier job attribute) that is unique for the system.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-comment
Input Synonym	comment
Type	Settable with the pdcreate or pdset commands.
Usage	
Explanation	Provides information that is associated with this print job.
Value Type	Single value.
Values	A text string up to 4095 characters that provides some information about the job.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-discard-time
Input Synonym	discard-time
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies the time and calendar date or time at which this print job is to be discarded, whether or not it has printed.
Value Type	Single value.
Values	Local time format; USA format is "HH:MM:SS mm/dd/yy" or HH:MM:SS. The string must be enclosed in quotes if the date is used.
Default	No default.
Usage Guidelines	

Attribute	Description
Name	job-end-message
Input Synonym	end-message
Type	Settable with the pdcreate or pdset commands.
Explanation	Provides the message that is to be sent to an operator when this job completes printing.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information on how the printed output is to be handled, such as special delivery instructions.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Validate and schedule against the printer attribute end-message-supported. o This message is sent if the physical printer attribute end-message-supported is "true" for the physical printer the job is submitted to. o The physical printer attribute notify-operator determines which operators receives the message.
Attribute	Description
Name	job-finishing
Type	Settable
Explanation	Identifies the desired finishing process to be applied to the printed output.
Value Type	Single value.
Values	Fixed Value none staple top-left-staple
Default	none.
Usage Guidelines	<ul style="list-style-type: none"> o Used to request a finishing process to be applied to the printed output.
Attribute	Description
Name	job-hold
Input Synonym	hold
Type	Settable with the pdcreate or pdset commands.

Explanation	Indicates whether a print job can be scheduled for printing.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	No default.						
Usage Guidelines	<ul style="list-style-type: none"> o If true, the job-state of the job is set to hold and the job-state-reasons is set to job-hold-set. o If held, the job remains in the queue until: <ul style="list-style-type: none"> - job-hold is set to "false" and the job can continue. - The time specified in the job-discard-time attribute is reached. 						
Attribute	Description						
Name	job-message-to-operator						
Input Synonym	message-to-operator						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Provides a message that is to be sent to an operator when a job is added to a queue.						
Value Type	Single value.						
Values	A text string up to 4095 characters that contains information about this job such as job processing requirements or some type of special handling.						
Default	No default.						
Usage Guidelines	<ul style="list-style-type: none"> o The person that is to receive this message is identified by the queue attribute notify-operator. 						
Attribute	Description						
Name	job-name						
Input Synonym	name						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Provides an identifier that may be printed on the job start sheet.						
Value Type	Single value.						
Values	A text string up to 255 characters that contains human readable job identification.						

Default	No default.
Usage Guidelines	o This identification may be printed on the banner page.
Attribute	Description
Name	job-originator
Input Synonym	originator
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the user ID of the person who submits a job.
Value Type	Single value.
Values	A text string up to 255 characters that contains the user ID. This value is set automatically by the DPS client program so this attribute is not used by a job that uses this initial-value-job.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-owner
Input Synonym	owner
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the name or user ID of the person responsible for this print job.
Value Type	Single value.
Values	A text string up to 255 characters that contains the global ID of a person that is to be responsible for this job.
Default	No default.
Usage Guidelines	o Use this attribute if you want to specify a person to be responsible for a job other than the person who submits the job. o This identification may be printed on the auxiliary-sheet page.
Attribute	Description
Name	job-print-after

Input Synonym	print-after
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies the time or time and calendar date after which the print job may be scheduled for printing.
Value Type	Single value.
Values	Local time format; USA format is HH:MM:SS or "HH:MM:SS mm/dd/yy". The string must be enclosed in quotes if the date is used.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job-state for this job is set to held and the job-state-reason is set to job-print-after-specified.
Attribute	Description
Name	job-priority
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies a number representing the scheduling priority for the job. This attribute is used by queues that employ a priority based scheduler. A higher value specifies a higher priority.
Value Type	Single value.
Values	An integer from 1 to 100.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This value should not be set higher than "50" for initial-value-job objects to be used by all users. If a value greater than 50 is specified, the value is lowered to "50" when the job is submitted. o A job submitted by an administrator can have any valid value.
Attribute	Description
Name	job-retention-period
Input Synonym	retention-period
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies the period of time following job completion that a job is to be retained before being discarded by the spooler.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes; hours are optional.

Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-sheets
Input Synonym	job-sheets
Type	Settable with the pdpr, pdset or pdmod commands.
Usage	Values vary by printer model.
Explanation	Specifies the auxiliary sheets that are inserted into the job.
Value Type	Single value.
Values	One of the following fixed values: none job-separators job-set-start job-set-end job-set-wrap job-set-start-copies-separate job-set-end-copies-separate job-set-wrap-copies-separate job-copy-start job-copy-end job-copy-wrap
Default	No default
Usage Guidelines	o The job is validated and scheduled against the printer attribute job-sheets-supported.
Attribute	Description
Name	job-start-message
Input Synonym	start-message
Type	Settable with the pdcreate or pdset commands.
Explanation	Provides a message that is to be sent to an operator when a job is sent to a physical printer conveying information about job processing.
Value Type	Single value.
Values	A text string up to 4095 characters that provides some information about the job such as this job is very important.
Default	No default.

Usage Guidelines	<ul style="list-style-type: none"> o Validate and schedule against the printer attribute start-message-supported. o This message is sent if the physical printer attribute start-message-supported is "true" for the physical printer the job is submitted to. o The persons that are to receive this message are identified in the physical printer attribute notify-operator.
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the persons responsible for the maintenance of this initial-value-job object.
Value Type	Multiple values.
Values	A text string up to 255 characters per person that contains the name or user ID of the person responsible.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is used by the initial-value-job object.
Attribute	Description
Name	logical-printers-ready
Type	Non Settable.
Explanation	Lists the enabled logical printers that reference this initial-value-job object.
Value Type	Multiple values.
Values	Set by HPDPS, it contains the global ID of the logical printer that is enabled or disabled and references this initial-value-job object.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Values are added when a logical printer is enabled and references this initial-value-job through its printer-initial-value-job attribute. o Values are removed when logical printers that reference this initial-value-job are disabled. o This initial-value-job object cannot be deleted until all printers have been removed from this list.
Attribute	Description

Name	message
Type	Settable with the pdcreate or pdset commands.
Explanation	Provides a message associated with the initial-value-job object.
Value Type	Single value.
Values	A text string up to 4095 characters that provides information about this object.
Default	No default.
Usage Guidelines	o This attribute is used by the initial-value-job object.
Attribute	Description
Name	notification-profile
Type	Settable with the pdcreate or pdset commands.
Explanation	Designates the people that are to be notified when specified events relating to this job and how the people are to be notified.
Value Type	Multiple values.
Values	<p>This is a complex attribute which has the following components:</p> <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale
Syntax	<pre>-x "notification-profile={event-identifiers=value delivery-method=value event-comment='text' delivery-address=value locale=value}"</pre> <p>For example,</p> <pre>-x "notification-profile={event-identifiers= class-state-changed job-completed delivery-method=electronic-mail event-comment='job progressing' delivery-address=joe@newhope locale=C}"</pre>
Component	Description
event-identifiers	<p>Explanation: Specifies the events for which the user is to receive notification that something has taken place.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the values listed for spooler attribute events-supported in pd_att_spooler.</p> <p>Default: No default.</p>

delivery-method	<p>Explanation: Specifies if or how the user is to receive the information.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>electronic-mail</td><td>e-mail, email</td></tr> <tr> <td>file</td><td></td></tr> <tr> <td>file-add-to</td><td></td></tr> <tr> <td>message</td><td></td></tr> <tr> <td>none</td><td></td></tr> </table> <p>Default: electronic-mail</p>	Fixed Value	Input Synonym	electronic-mail	e-mail, email	file		file-add-to		message		none	
Fixed Value	Input Synonym												
electronic-mail	e-mail, email												
file													
file-add-to													
message													
none													
event-comment	<p>Explanation: This textual information is appended to the event message.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that supplies additional information.</p> <p>Default: No default.</p>												
delivery-address	<p>Explanation: The address of the person that is to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the name and node of the person that is to receive the information.</p> <p>Default: No default.</p>												
locale	<p>Explanation: Identifies the locale of the person that is to receive the message.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the locale identification of the person that is to receive the notification. This information is used to determine the language and coded character set that the notification is sent in.</p> <p>Default: No default.</p>												
Attribute	Description												
Name	object-class												
Type	Non Settable												
Explanation	Identifies the object class to which this object belongs.												
Value Type	Single value.												
Values	The value of this attribute is initial value job.												
Default	initial-value-job												
Usage Guidelines	o This attribute is used by the initial-value-job object.												
Attribute	Description												
Name	physical-printers-requested												
Type	Settable with the pdcreate or pdset commands.												

Explanation	Identifies the physical printers requested for printing the job.
Value Type	Multiple values.
Values	A text string up to 255 characters per printer that contains the name of the physical printer.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job will be printed on one of the requested physical printers, if possible. o Validate and schedule against the printer attribute printer-name. o Validate and schedule against the printer attribute start-message-supported. o This message is sent if the physical printer attribute start-message-supported is "true" for the physical printer the job is submitted to. o The persons that are to receive this message are identified in the physical printer attribute notify-operator.
Attribute	Description
Name	printer-locations-requested
Input Synonym	locations-requested
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the locations where a job has been requested to be printed.
Value Type	Multiple values.
Values	A text string up to 4095 characters per printer that contains the physical location of the printer being requested.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job will be printed on a printer at one of the requested locations, if possible. o The job is validated and scheduled against the printer attribute printer-locations. o Any individual value specified for this attribute can select more than one physical printer. For example, the printers could all be in the same location, such as a printer room. o The values for both the job and printer attributes are text strings that can include blanks. They must match exactly for validation to occur.
Attribute	Description
Name	printer-models-requested

Input Synonym	models-requested
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies the model IDs of the printers on which the job is to be printed.
Value Type	Multiple values.
Values	A text string up to 4095 characters per printer that contains the model of the printer that can be used to print this job.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job will be printed on a requested model, if possible. o The job is validated and scheduled against the printer-attribute printer-model. o Any individual value specified for this attribute can select more than one physical printer.
Attribute	Description
Name	printer-types-requested
Input Synonym	types-requested
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Identifies the type of the printer to be used for printing the job.
Value Type	Multiple values
Values	<p>Any of the following fixed values:</p> <p>other electrographic-led electrographic-laser electrographic-other impact-moving-head-dot-matrix-9-pin impact-moving-head-dot-matrix-24-pin impact-moving-head-dot-matrix-other impact-moving-head-fully-formed impact-band impact-other inkjet-aqueous inkjet-solid inkjet-other pen thermal-transfer thermal-sensitive thermal-diffusion thermal-other electro-erosion electro-static photographic-microfiche photographic-imagesetter</p>

	photographic-other ion-deposition E-beam typesetter
Default	No default
Usage Guidelines	o The job is validated and scheduled against the printer attribute printer-types.
Attribute	Description
Name	results-profile
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Specifies the delivery method for the hardcopy output, and designates who is to receive output and the number of copies (per recipient).
Value Type	Multiple values.
Values	This is a complex attribute which has the following components: 1. delivery-method 2. results-set-comment 3. delivery-address 4. job-copies 5. output-bin
Syntax	-x "results-profile=\ dm value:rc value:da value:jc value:ob value" For example: -x "::::2:top"
Usage Guidelines	o Only job-copies and output-bin components are used.
Component	Description
delivery-method	Explanation: Specifies how the person is to receive the job output. Value Type: Single value. Values: The only value for this component is the following fixed value: pickup Default: pickup
results-set-comment	Explanation: Supplies a text string that describes the results-set value. Value Type: Single value. Values: A text string up to 4095 characters that provides information such as "Please staple this document". Or information such as "Please read before meeting tomorrow" for the person that is to receive the document. This information may be printed on start-sheets. Default: No default.
delivery-	Explanation: The address of the person that is to

address	<p>receive the job output.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the address.</p>
job-copies	<p>Explanation: Specifies how many copies of the job the person is to receive.</p> <p>Value Type: Single value.</p> <p>Values: An integer (cannot be 0).</p> <p>Default: 1</p>
output-bin	<p>Explanation: Specifies the output-bin to be used. The job is validated against the physical printer attribute output-bins-supported.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <p>top</p> <p>middle</p> <p>bottom</p> <p>side</p> <p>left</p> <p>right</p> <p>face-up</p> <p>face-down</p> <p>large</p> <p>private</p> <p>collator</p> <p>Default: No default.</p> <p>The job is validated against the printer attribute output-bins-supported.</p>

NAME

pd_att_job - list of attributes for an HPDPS job object

DESCRIPTION

The following is a list of valid attributes and values for the job object class of the HP Distributed Print Service.

Attributes for a Job Object

Settable Attribute Listing	Specifiable Attribute Listing
interface-program-method job-batch job-comment job-discard-time job-end-message job-finishing job-hold job-message-from-administrator job-message-to-operator job-name job-print-after job-priority job-retention-period job-sheets job-start-message notification-profile physical-printers-requested printer-locations-requested printer-models-requested printer-mopy-requested printer-types-requested results-profile	initial-value-job job-originator job-owner printer-name-requested

Table 1-1. Job Attributes

Attribute	Description
Name	client-basic-ds
Type	Non Settable
Explanation	Identifies the host to contact for information about the source environment from which the job was submitted. This attribute is present only when the job is submitted to an HPDPS Gateway printer.
Value Type	Single value.
Values	A text string containing the IP address of the host in dotted decimal format.
Default	No default.
Usage Guidelines	The destination environment uses this attribute to contact the source environment, such as to perform dce-pipe-pull document transfers. The IP address is of a host that runs a basicdsd server for the source environment.

Attribute	Description
Name	completion-time
Type	Non Settable
Explanation	Identifies the time when the job completed printing. Set by HPDPS.
Value Type	Single value.
Values	Local time format; USA is "HH:MM:SS mm/dd/yy"
Default	No default.
Usage Guidelines	Provides status information.
Attribute	Description
Name	current-job-state
Input Synonym	job-state
Type	Non Settable
Explanation	Identifies the current state of the job.
Value Type	Single value.
Values	One of the following fixed values: canceled held paused pending pre-processing processing retained terminating timed-out unknown
Default	No default.
Usage Guidelines	o The value is set and updated by HPDPS as the job progresses.
Attribute	Description
Name	device-name-assigned
Type	Non Settable.
Usage	
Explanation	Identifies the path name of the device special file

	associated with the physical printer.
Value Type	Single Value.
Values	A text string that contains the path name of the device associated with the printer. Implicitly set by HPDPS if the value of printer attribute attachment-type is serial or parallel.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is intended for use in the ipmap file to pass the device path name to the interface program that accesses the device.
Attribute	Description
Name	initial-value-job
Type	Specifiable with the pdpr command.
Explanation	Identifies the initial-value-job object in the spooler that is to be used to create the job object during job submission.
Value Type	Single value.
Values	The name of an existing initial-value-job object.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o If a value is specified for this attribute, the job uses the attributes and values from the object unless those are overridden by attribute values supplied at the command line.
Attribute	Description
Name	interface-program-method
Input Synonym	ip-method
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies the invocation method of the interface program.
Value Type	Single value.
Values	One of the following fixed values: hpdps lp
Default	hpdps
Usage Guidelines	<ul style="list-style-type: none"> o The job is validated against the printer attribute interface-program-methods-supported. o If the value hpdps is specified, the interface program is invoked to print one job result-set as follows:

	<ul style="list-style-type: none"> - The value of the job-copies component of the job results-profile attribute is passed as the 4th argument of the interface program. - Each document is duplicated n times before being passed to the interface program, where n is the value of the document copy-count attribute. o If the value lp is specified, the interface program is invoked to print each job copy in one job result set: <ul style="list-style-type: none"> - The value of document attribute copy-count is passed as the 4th argument of the interface program. - The interface program is invoked n times, where n is the value of the job-copies component of the job results-profile attribute.
Attribute	Description
Name	intervening-jobs
Input Synonym	position-in-queue, queue-position
Type	Non Settable
Explanation	Indicates the number of jobs in the queue before this job.
Value Type	Single value.
Values	An integer from 0 to 2147483647. Set and updated by HPDPS.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Provides status information. o A value of zero (0) means that the job is currently printing.
Attribute	Description
Name	job-batch
Type	Settable
Explanation	Specifies that the job you are submitting is being marked as a specific type of job. HPDPS processes this job only on a physical printer which is ready to print that specified job-batch type.
Value Type	Single value.
Values	You can enter a value up to 4095 characters that contains the job-batch name.
Default	No default.
Usage Guidelines	HPDPS schedules this attribute against the physical printer's job-batches-ready attribute.
Attribute	Description

Name	job-client-id
Input Synonym	local-id
Type	Non Settable
Explanation	Identifies the local job identifier.
Value Type	Single value.
Values	The local ID. HPDPS sets this value from 1 through the number specified as the value for the PDIDTABLE environment variable.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> Generated by the client program and is unique for a given user ID. The local ID is mapped to global ID (the job attribute job-identifier), which is unique for the system.
Attribute	Description
Name	job-comment
Input Synonym	comment
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Provides information that is associated with this job.
Value Type	Single value.
Values	A text string up to 4095 characters that contains some information associated with this job.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-copies-completed
Type	Non Settable
Explanation	Indicates the total number of copies of the job that have finished printing.
Value Type	Single value.
Values	An integer from 0 to 2147483647. Set and updated by HPDPS.
Default	No default.

Usage Guidelines	o This value is updated at the completion of each result-set.
Attribute	Description
Name	job-discard-time
Input Synonym	discard-time
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies the time or time and calendar date at which the print job is to be discarded, whether or not it has printed.
Value Type	Single value.
Values	Local time format: USA format is HH:MM:SS or "HH:MM:SS mm/dd/yy" The string must be quoted if the date is specified.
Default	No default
Usage Guidelines	
Attribute	Description
Name	job-end-message
Input Synonym	end-message
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Provides a message conveying information about output handling when the job completes printing.
Value Type	Single value.
Values	A text string up to 4095 characters that contains instructions to an operator such as special output delivery instructions.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Validate and schedule the job against the printer attribute end-message-supported. o This message is sent if the attribute end-message-supported is "true" for the physical printer to which the job is submitted. o The printer attribute notify-operator identifies the operators that are to receive the message.
Attribute	Description
Name	job-finishing

Type	Settable						
Explanation	Identifies the desired finishing process to be applied to the printed output.						
Value Type	Single value.						
Values	Fixed Value none staple top-left-staple						
Default	none.						
Usage Guidelines	<ul style="list-style-type: none"> o Used to request a finishing process to be applied to the printed output. 						
Attribute	Description						
Name	job-hold						
Input Synonym	hold						
Type	Settable with the pdpr, pdset or pdmod commands.						
Explanation	Indicates whether the print job can be scheduled for printing.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	false						
Usage Guidelines	<ul style="list-style-type: none"> o If true, the job-state changes to hold and the job-state-reasons is set to job-hold-set. o If held, the job remains in the queue until: <ul style="list-style-type: none"> - job-hold is set to "false" and the job can continue. - The time specified in the job-discard-time attribute is reached. 						
Attribute	Description						
Name	job-identifier						
Input Synonym	global-id						
Type	Non Settable						
Explanation	The global job identifier. It uniquely identifies the job within the server.						
Value Type	Single value.						

Values	The global ID. Set by HPDPS.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-message-from-administrator
Input Synonym	message-from-administrator
Type	Settable; see Usage Guidelines.
Explanation	Describes the reasons that the job is being or has been changed.
Value Type	Single value.
Values	A text string up to 4095 characters that provides information concerning why a given action has been taken.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Set with the value specified with the -m flag or the command attribute message when included: <ul style="list-style-type: none"> - With the pdmod or pdset command and the object class is job. - With the pdpause, pdresume, pdpromote, or pdrm commands.
Attribute	Description
Name	job-message-to-operator
Input Synonym	message-to-operator
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Provides a message that is to be sent to an operator when the job is added to a queue.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information about job processing requirements such as some type of special handling.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The operator that is to receive the message is identified in the queue attribute notify-operator.
Attribute	Description

Name	job-name
Input Synonym	name
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Provides a human readable job identification.
Value Type	Single value.
Values	A text string up to 255 characters that contains the identification.
Default	The file name of the first document in the job.
Usage Guidelines	o This identification may be printed on the banner page.
Attribute	Description
Name	job-originator
Input Synonym	originator
Type	Specifiable with the pdpr command.
Explanation	Identifies the user ID, or user ID and node, of the person who submitted the job, or the name of the program that initiated the job.
Value Type	Single value.
Values	A text string up to 255 that contains the global ID.
Default	username@node of person submitting the job.
Usage Guidelines	
Attribute	Description
Name	job-owner
Input Synonym	owner
Type	Specifiable with the pdpr command.
Explanation	Identifies the name or user ID of the person responsible for the print job.
Value Type	Single value.
Values	A text string up to 255 that contains the global ID of person that is to be responsible for this job.
Default	The value of the job attribute job-originator.

Usage Guidelines	<ul style="list-style-type: none"> o Use this attribute if you want to specify a person to be responsible for the job other than yourself. o This identification may be printed on the auxiliary-sheet.
Attribute	Description
Name	job-print-after
Input Synonym	print-after
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies the time or time and calendar date after which the print job may be scheduled for printing.
Value Type	Single value.
Values	Local time format; for USA HH:MM:SS or "HH:MM:SS mm/dd/yy". The string must be quoted if the date is specified.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o If submitted before the specified time, job-state is set to held and the job-state-reasons is identified as job-print-after-specified for the job.
Attribute	Description
Name	job-priority
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies a number representing the scheduling priority for the job. This attribute is used by queues that employ a priority based scheduler. A higher value specifies a higher priority.
Value Type	Single value.
Values	An integer from 1 to 100.
Default	50
Usage Guidelines	<ul style="list-style-type: none"> o A job submitter should not set this value higher than "50" unless the job submitter is an administrator. If a value greater than 50 is specified, the value is set to "50". o An administrator can set the value to any valid value.
Attribute	Description
Name	job-retention-period
Input	retention-period

Synonym	
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Specifies the period of time following job completion that the job is to be retained before being discarded by the spooler.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes; hours are optional.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-sheets
Type	Settable with the pdpr, pdset or pdmod commands.
Usage	Values vary by printer model.
Explanation	Specifies the auxiliary sheets that are inserted into the job.
Value Type	Single value.
Values	One of the following fixed values: none job-separators job-set-start job-set-end job-set-wrap job-set-start-copies-separate job-set-end-copies-separate job-set-wrap-copies-separate job-copy-start job-copy-end job-copy-wrap
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o The job is validated and scheduled against the printer attribute job-sheets-supported. <p>Most printer models support the following 2 values:</p> <ul style="list-style-type: none"> o none No auxiliary sheets, or banner pages, are printed with the job. o job-set-start A banner page is printed at the start of each job-result-set. For most printers, this means a single banner page is printed before the job documents are printed.

Attribute	Description
Name	job-start-message
Input Synonym	start-message
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Provides a message that is to be sent to an operator when the job is added to a queue conveying information about job processing.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information about the job such as, "This job is very important; call me if there any problem."
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Validate and schedule the job against the printer attribute start-message-supported. o A message is sent if the physical printer attribute start-message-supported is "true" for the physical printer to which the job is submitted. o The printer attribute notify-operator contains the identity of the operators that are to receive the message.
Attribute	Description
Name	job-state-reasons
Input Synonym	state-reasons, reasons
Type	Non Settable
Explanation	Identifies the reasons that a job is in the held, terminating, or retained state. If this attribute value is blank, the job is not in one of these states.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <ul style="list-style-type: none"> aborted-by-system canceled-by-operator canceled-by-user completed-with-errors completed-with-warnings completed-successfully job-hold-set job-print-after-specified required-resources-not-ready required-resources-not-supported
Default	No default.

Usage Guidelines							
Attribute	Description						
Name	job-submission-complete						
Type	Non Settable						
Explanation	Indicates whether all documents of the job have been received by the spooler.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	No default.						
Usage Guidelines							
Attribute	Description						
Name	modification-time						
Type	Non Settable						
Explanation	Identifies the time this job was last modified.						
Value Type	Single value.						
Values	Local time format. For the USA; "HH:MM:SS mm/dd/yy"						
Default	No default.						
Usage Guidelines							
Attribute	Description						
Name	name-of-last-accesser						
Input Synonym	last-accesser, last-modifier						
Type	Non Settable						
Explanation	Identifies the person or name of the program that submitted the job, or most recently modified the job. If the job was submitted or modified by a person, the value is the login ID for that user.						
Value Type	Single value.						
Values	A text string up to 255 characters that contains the login ID of the user or a global ID. Set by HPDPS.						

Default	No default.
Usage Guidelines	
Attribute	Description
Name	new-job-identifier
Input Synonym	new-identifier, new-id
Type	Non Settable
Explanation	Identifies the name of the spooler to which this job has been reassigned. Also a new global job identifier, if a job was resubmitted to a logical printer on a different spooler.
Value Type	Single value.
Values	server-name:job-identifier or server-name. Set by HPDPS.
Default	No default.
Usage Guidelines	o If the job is submitted to a logical printer on the same spooler, the job identifier remains the same.
Attribute	Description
Name	notification-profile
Type	Settable with the pdpr, pdset or pdmod commands.
Explanation	Designates which persons are to be notified of specific events relating to this job, and how the persons are to be notified.
Value Type	Multiple Values
Values	This is a complex attribute, which has the following components: <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale
Syntax	<pre>-x "notification-profile={event-identifiers=value delivery-method=value event-comment='text' delivery-address=value locale=value}"</pre> <p>For example:</p> <pre>-x "notification-profile={event-identifiers=class-error delivery-method=electronic-mail event-comment='fix problem' delivery-address=mary@travel locale=C}"</pre>

Component	Description												
event-identifiers	<p>Explanation: Specifies the events for which the user is to receive messages.</p> <p>Value Type: Multiple values.</p> <p>Values: A text string up to 4095 characters that includes any of the values listed for the attribute events-supported of a spooler (pd_att_spooler) or supervisor (pd_att_supervisor).</p> <p>Default:</p> <ul style="list-style-type: none"> document-aborted-by-printer document-aborted-by-server document-canceled-at-printer job-aborted-by-server job-canceled-by-operator job-cannot-be-scheduled job-completed job-discarded printer-needs-attention printer-needs-operator printer-paper-jam printer-paper-out printer-paper-output-problem printer-toner-low 												
delivery-method	<p>Explanation: Specifies how the person is to receive event messages.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table border="0"> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>electronic-mail</td><td>e-mail, email</td></tr> <tr> <td>file</td><td></td></tr> <tr> <td>file-add-to</td><td></td></tr> <tr> <td>message</td><td></td></tr> <tr> <td>none</td><td></td></tr> </table> <p>Default: electronic-mail</p>	Fixed Value	Input Synonym	electronic-mail	e-mail, email	file		file-add-to		message		none	
Fixed Value	Input Synonym												
electronic-mail	e-mail, email												
file													
file-add-to													
message													
none													
event-comment	<p>Explanation: Supplies textual information that is appended to the event message.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that supplies additional information concerning the event.</p> <p>Default: No value.</p>												
delivery-address	<p>Explanation: The address of the person to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the name and node of the person that is to receive notification.</p> <p>Default: The login of the user that submitted the job using the pdpr command.</p>												
locale	<p>Explanation: Identifies the locale of the user that is to receive the messages.</p> <p>Value Type: Single value.</p> <p>Values: A text string that identifies the locale to be used. This is used to determine the language and coded character set that the information is to be sent in.</p> <p>Default: The locale of the user that submitted the job with the pdpr command.</p>												

Attribute	Description
Name	number-of-documents
Type	Non Settable
Explanation	Identifies the number of documents in the job, including resource documents such as fonts.
Value Type	Single value.
Values	An integer from 1 to 2147483647. Set by HPDPS.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is job.
Default	job
Usage Guidelines	
Attribute	Description
Name	octets-completed
Type	Non Settable
Explanation	Reports the number of octets (bytes) in this job that have been printed.
Value Type	Single value.
Values	An integer from 0 to 9223372036854775800. Set by HPDPS.
Default	No default.
Usage Guidelines	o This attribute is updated at the completion of each result-set.
Attribute	Description

Name	output-bin-requested
Type	Non Settable
Explanation	Identifies the output-bin specified by the output-bin component of the job results-profile attribute.
Value Type	Single Value.
Values	<p>A text string of up to 255 characters that describes the output bin, or any of the following fixed values:</p> <p>top middle bottom side left right face-up face-down large private collator</p> <p>Implicitly set by HPDPS.</p>
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is intended for use in the ipmap file to convert the output-bin component of the job results-profile attribute to lp options.
Attribute	Description
Name	pages-completed
Type	Non Settable
Explanation	Reports the number of pages in this job that have been printed.
Value Type	Single value.
Values	An integer from 0 to 2147483647. Set by HPDPS.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute may or may not be updated depending on the requested physical printer.
Attribute	Description
Name	physical-printers-requested
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Identifies the physical printers that are being requested to print this job.
Value Type	Multiple values.

Values	A text string up to 255 characters per physical printer that contains the name of the physical printer.
Default	No default.
Usage Guidelines	o The job is validated and scheduled against the printer attribute printer-name.
Attribute	Description
Name	previous-job-state
Input Synonym	previous-state
Type	Non Settable
Explanation	Identifies the state of the job before the last job state change.
Value Type	Single value.
Values	One of the following fixed values set by HPDPS: canceled i held paused pending pre-processing processing retained terminating timed-out unknown
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-initial-value-job
Type	Non Settable
Explanation	Associates an initial-value-job object with the logical printer, which this job was submitted to.
Value Type	Single value.
Values	Set by HPDPS.
Default	No default.
Usage Guidelines	o The associated initial-value-job object is used to supply values for attributes of jobs submitted to this logical printer. These values override server defaults but are overridden by job attribute values

	specified on the command line.
Attribute	Description
Name	printer-locations-requested
Input Synonym	locations-requested
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Identifies the locations where the job is being requested to be printed.
Value Type	Multiple values.
Values	A text string up to 4095 characters that contains the physical location of each printer requested. The job will be printed on a printer at one of the locations if the validation is successful.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job is validated and scheduled against the printer attribute printer-locations. o Any individual value specified for this attribute can select more than one physical printer. For example, the printers could all be in the same location, such as a printer room. o The values for both the job and printer attributes are text strings that can include blanks. They must match exactly for validation to occur.
Attribute	Description
Name	printer-models-requested
Input Synonym	models-requested
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Identifies the model ID of the printers requested to print the job.
Value Type	Multiple values.
Values	A text string up to 4095 characters per printer that contains the printer model to be used to print the job.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job is validated and scheduled against the printer attribute printer-model. Only one value must match for the job to be validated and scheduled. o Any individual value specified for this attribute can select more than one physical printer.
Attribute	Description

Name	printer-mopy-requested
Type	Settable
Explanation	Requests to print multiple copies by using printer's internal disk spooling capability.
Value Type	Single boolean value.
Values	Fixed Value true false
Default	false.
Usage Guidelines	More relevant when submitting multiple copies of large jobs, this feature, when supported by physical printer, relieves HPDPS from resubmitting a job to the printer.
Attribute	Description
Name	printer-types-requested
Input Synonym	types-requested
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Identifies the type of the printer to be used for printing the job.
Value Type	Multiple values
Values	Any of the following fixed values: other electrographic-led electrographic-laser electrographic-other impact-moving-head-dot-matrix-9-pin impact-moving-head-dot-matrix-24-pin impact-moving-head-dot-matrix-other impact-moving-head-fully-formed impact-band impact-other inkjet-aqueous inkjet-solid inkjet-other pen thermal-transfer thermal-sensitive thermal-diffusion thermal-other electro-erosion electro-static photographic-microfiche photographic-imagesetter photographic-other ion-deposition E-beam

	typesetter
Default	No default
Usage Guidelines	o The job is validated and scheduled against the printer attribute printer-types.
Attribute	Description
Name	printer-name-requested
Input Synonym	printer-requested, logical-printer-requested
Type	Specifiable with the pdpr command.
Explanation	Identifies the logical printer to which this job is to be submitted.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of the logical printer.
Default	The PDPRINTER environment variable for the client.
Usage Guidelines	o To move the job to a different logical printer, use the pdresubmit command.
Attribute	Description
Name	printers-assigned
Type	Non Settable
Explanation	Identifies the physical printer to which the job has been assigned for printing.
Value Type	Single value.
Values	The global ID of the printer.
Default	No default.
Usage Guidelines	o If the value for the attribute is blank, the job is still waiting to be scheduled to a physical printer.
Attribute	Description
Name	printers-used
Type	Non Settable
Explanation	Identifies the physical printers where this job was printed.
Value Type	Multiple values.
Values	Value set by HPDPS.

Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> Normally this value only contains one physical printer name. However, if a job had to be restarted on another printer because the first printer failed for some reason, this value could contain more than one physical printer name.
Attribute	Description
Name	processing-time
Type	Non Settable
Explanation	Reports the amount of time that the job has been printing on the physical printer device. If printing has completed, this value is the total amount of time needed to print the job.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes. Set by HPDPS.
Default	No default.
Usage Guidelines	This attribute is updated when a user queries the job with the pdls command. The value is calculated by comparing the value of the started-printing-time attribute of job with the current time.
Attribute	Description
Name	queue-assigned
Type	Non Settable
Explanation	Identifies the queue to which the job has been assigned.
Value Type	Single value.
Values	A text string up to 255 characters set by HPDPS that contains the name of the queue.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> If this attribute is blank, the job is not currently residing in a queue (it may be in the retained or timed-out state).
Attribute	Description
Name	results-profile
Type	Settable with the pdpr, pdset, or pdmod commands.
Explanation	Specifies the delivery method for the hardcopy output, and designates who is to receive output and the number of copies (per recipient).

Value Type	Multiple values.
Values	<p>This is a complex attribute which has the following components:</p> <ol style="list-style-type: none"> 1. delivery-method 2. results-set-comment 3. delivery-address 4. job-copies 5. output-bin
Syntax	<pre>-x "results-profile=\ dm value:rc value:da value:jc value:ob value"</pre> <p>For example: -x "::::2:top"</p>
Usage Guidelines	<ul style="list-style-type: none"> o Only the job-copies and output-bin components are used by HPDPS.
Component	Description
delivery-method	<p>Explanation: Specifies how the person is to receive the job output.</p> <p>Value Type: Single value.</p> <p>Values: The only value for this component is the following fixed value:</p> <p>pickup</p> <p>Default: pickup</p>
results-set-comment	<p>Explanation: Supplies a text string that describes the results-set value.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that provides information such as "Please staple this document". Or information such as "Please read before meeting tomorrow" for the person that is to receive the document. This information may be printed on start-sheets.</p> <p>Default: No default.</p>
delivery-address	<p>Explanation: The address of the person that is to receive the job output.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the address.</p>
job-copies	<p>Explanation: Specifies how many copies of the job the person is to receive.</p> <p>Value Type: Single value.</p> <p>Values: An integer (cannot be 0).</p> <p>Default: 1</p>
output-bin	<p>Explanation: Specifies the output-bin to be used.</p> <p>Value Type: Single value.</p> <p>Values: A text string of up to 255 characters that describes the output bin, or any of the following fixed values:</p> <p>top middle bottom side</p>

	left right face-up face-down large private collator Default: No default. The value is validated against the printer attribute output-bins-supported.
Attribute	Description
Name	started-printing-time
Type	Non Settable
Explanation	Identifies the time the job was sent to the physical printer.
Value Type	Single value.
Values	Local time format; USA format is "HH:MM:SS mm/dd/yy". Set by HPDPS.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	submission-time
Type	Non Settable
Explanation	Identifies the time the job was added to the queue.
Value Type	Single value.
Values	Local time format; for USA, "HH:MM:SS mm/dd/yy". Set by HPDPS.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	total-job-octets
Input Synonym	job-size
Type	Non Settable
Explanation	The size of the job determined by the sum total of all

	printable octets in the job.
Value Type	Single value.
Values	An integer from 0 to 9223372036854775800. Set by HPDPS.
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o The server computes this value by totaling the size of all printable documents. It is based on the number of printable octets and is computed as: job-copies X (printable document octet-count X copy-count) o The job is validated against the logical printer attribute job-size-range-supported. o The job is scheduled against the physical printer attribute job-size-range-ready.
Attribute	Description
Name	user-locale
Type	Non Settable
Explanation	Identifies the language of the information that appears on the start, separator, and end sheets printed with the job and the messages for the job that are sent to the various people.
Value Type	Single value.
Values	The locale of the job submitter
Default	Set from the operation-locale, which is set from the user-locale.
Usage Guidelines	
Attribute	Description
Name	user-name
Type	Non Settable
Explanation	Identifies the login ID of the job submitter.
Value Type	Single value.
Values	A text string that contains the login of the job submitter; username@node.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Set by HPDPS to username@node.

NAME

pd_att_log - list of attributes for an HPDPS log object

DESCRIPTION

The following is a list of valid attributes and values for the log object class of the HP Distributed Print Service.

Log objects are automatically created when the server is initialized. These logs contain and keep track of such things as error messages and trace messages. These logs cannot be created or deleted, but they can be enabled or disabled. A configuration file can be used to set the initial values of attributes for error and trace logs when servers are initialized.

The following are restrictions for log attributes:

- The **log-severity** attribute is an attribute only for error logs.
- The **log-trace-groups** attribute is an attribute only for trace logs. TP • The values for the attributes listed below cannot be set with the pdset command for trace logs. Values can be specified using configuration files during server initialization.

log-address
log-size
log-wrap

Attributes for a Log Object

Settable Attribute Listing	Specifiable Attribute Listing
descriptor list-of-managers log-address log-severity log-size log-trace-groups log-wrap message	log-type

Table 1-1. Log Attributes	
Attribute	Description
Name	associated-server
Type	Non Settable
Explanation	Indicates the name of the server in which this log resides.
Value Type	Single value.
Values	The name of the server; set by HPDPS when this log is created.
Default	No default.
Usage Guidelines	
Attribute	Description

Name	descriptor						
Type	Settable with the pdset command.						
Explanation	Provides a description of the object.						
Value Type	Single value.						
Values	A text string up to 4095 characters that describes this log.						
Default	If trace log: This is the standard trace log. If error log: This is the standard error log.						
Usage Guidelines							
Attribute	Description						
Name	enabled						
Type	Non Settable						
Explanation	Indicates if the log is enabled. An enabled log writes messages to a log file.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	If error log: true If trace log: false						
Usage Guidelines	o Implicitly set by the pdenable and pddisable commands.						
Attribute	Description						
Name	list-of-managers						
Input Synonym	managers						
Type	Settable with the pdset command.						
Explanation	Lists the people responsible for the maintenance and operation of this log.						
Value Type	Multiple value.						
Values	A text string up to 255 characters per person that contains the user ID of the person responsible for this log.						

Default	No default
Usage Guidelines	
Attribute	Description
Name	log-address
Type	Non Settable
Explanation	Specifies the pathname where the log resides.
Value Type	Single Value
Values	A text string up to 4095 characters that contains the pathname.
Default	The path defined by the PDBASE environment variable plus the ServerName (\$PDBASE/ServerName).
Usage Guidelines	o This attribute defines only the path.
Attribute	Description
Name	log-identifier
Type	Non Settable
Explanation	Uniquely identifies this log.
Value Type	Single Value
Values	A text string up to 255 characters set by HPDPS that contains the file name of this log.
Default	If trace log: default_tracelog If error log: default_errorlog
Usage Guidelines	
Attribute	Description
Name	log-severity
Type	Settable with the pdset command.
Explanation	Identifies the severity level of the messages that are to be logged in this error log.
Value Type	Single Value
Values	One of the following fixed values: error warning audit

	debug info
Default	audit
Usage Guidelines	o This attribute is only for error logs.
Attribute	Description
Name	log-size
Type	Non Settable for trace logs. Settable with the pdset for error logs.
Explanation	Specifies how large (kilobytes) a log file is allowed to get before the file wraps, or the logging of events is halted. Whether a log file wraps or logging is halted is determined by the value of the attribute log-wrap.
Value Type	Single Value
Values	An integer from 1 to 2147483647 (value represents kilobytes)
Default	If error log: 1024 If trace log: 8192
Usage Guidelines	o The value specified in the configuration file, if one is specified, will override the default value when the server is created. o If this attribute is changed, the current log file is renamed to error.log.BAK, and a new log file is created.
Attribute	Description
Name	log-trace-groups
Type	Settable with the pdset command.
Explanation	Specifies the active trace groups for the selective tracing of HPDPS component groups.
Value Type	Multiple values.
Values	Any text string up to 4095 characters that contains trace group IDs that are to be traced.
Default	???? -1 (Meaning all trace groups)
Usage Guidelines	o This attribute is only for trace logs and is only to be used at the request of an HP Service Representative.
Attribute	Description
Name	log-type

Type	Non Settable						
Explanation	Identifies the type of log.						
Value Type	Single Value						
Values	One of the following fixed values: errorlog tracelog						
Default	No default.						
Usage Guidelines							
Attribute	Description						
Name	log-wrap						
Type	Settable with the pdset command for error logs.						
Explanation	Specifies whether the log file should wrap or halt when the specified value in the log-size attribute is reached.						
Value Type	Single Value						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	true						
Usage Guidelines	o The value specified in the configuration file, if one is specified, will override the default value when the server is created.						
Attribute	Description						
Name	message						
Type	Settable with the pdset command.						
Explanation	Provides a message that is associated with this log object.						
Value Type	Single value.						
Values	A text string up to 4095 characters that contains information about this log.						
Default	No default.						
Usage Guidelines							

Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is log.
Default	log
Usage Guidelines	

NAME

pd_att_log_ptr - list of attributes for an HPDPS logical printer object

DESCRIPTION

The following is a list of valid attributes and values for the logical printer object class of HP Distributed Print Services.

Logical printers are used for job routing, defaulting, and job validation. If a logical printer attribute does not have a value, then the corresponding attribute for the physical printer is used. The default for most logical printer attributes is 'No value'. This means that the logical printer accepts any valid value for those attributes.

Attributes for a Logical Printer Object

Settable Attribute Listing	Specifiable Attribute Listing
associated-queue authorize-jobs character-sets-supported content-orientations-supported descriptor document-attributes-supported document-formats-supported document-types-supported end-messages-supported fonts-supported input-trays-supported interface-program-methods-supported job-attributes-supported job-finishings-supported job-sheets-supported job-size-range-supported list-of-managers maximum-copies-supported media-supported message notification-profile numbers-up-supported page-select-supported pd-gway-foreign-host pd-gway-foreign-printer plexes-supported print-qualities-supported printer-resolutions-supported printer-initial-value-document printer-initial-value-job printer-locations printer-model printer-mopy-supported sides-supported start-message-supported	printer-realization

Table 1-1. Logical Printer Attributes	
Attribute	Description
Name	associated-queue
Type	Settable with the pdcreate or pdset commands.

Explanation	Identifies the queue associated with this printer. All jobs submitted to this logical printer will be sent to the queue specified by this attribute.						
Value Type	Single value.						
Values	A text string up to 255 characters that contains this global ID of the queue.						
Default	No default.						
Usage Guidelines							
Attribute	Description						
Name	associated-server						
Type	Non Settable						
Explanation	Indicates the name of the spooler in which this printer resides.						
Value Type	Single value.						
Values	A text string up to 255 characters that contains the name of the server. Set from the argument (ServerName:PrinterName) of the pdcreate command when this logical printer was created.						
Default	No default.						
Usage Guidelines							
Attribute	Description						
Name	authorize-jobs						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Identifies if the person submitting the job is required to be authorized to submit a job to this logical printer.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	false						
Usage Guidelines	<ul style="list-style-type: none"> When the spooler security level is medium the following rules apply when the value for this attribute is: <ul style="list-style-type: none"> True The login ID of the job submitter is used to check if the person has "read" authority 						

	<p>on this printer. If the person does have "read" authority, the job is authorized.</p> <p>False The job is automatically authorized.</p> <p>Note: Security level of medium is the only supported level.</p>
Attribute	Description
Name	character-sets-supported
Type	Settable with pdcreate or pdset commands.
Explanation	Identifies the character set encodings supported by this printer.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <p>iso-ucs-2-level2</p> <p>other</p> <p>ascii</p> <p>iso-latin1</p> <p>iso-latin2</p> <p>iso-latin3</p> <p>iso-latin4</p> <p>iso-latin-cyrillic</p> <p>iso-latin-arabic</p> <p>iso-latin-greek</p> <p>iso-latin-hebrew</p> <p>iso-latin5</p> <p>iso-latin6</p> <p>iso-text-communication</p> <p>half-width-katakana</p> <p>jis-encoding</p> <p>shift-jis</p> <p>euc-packed-format-japanese</p> <p>euc-fixed-width-japanese</p> <p>iso-reg4-united-kingdom</p> <p>iso-reg11-swedish-for-names</p> <p>iso-reg15-italian</p> <p>iso-reg17-spanish</p> <p>iso-reg21-german</p> <p>iso-reg60-danish-norwegian</p> <p>iso-reg69-french</p> <p>unicode</p> <p>ucs4</p> <p>unicode-ascii</p> <p>unicode-latin1</p> <p>unicode-ibm-2039</p> <p>unicode-ibm-1261</p> <p>unicode-ibm-1268</p> <p>unicode-ibm-1276</p> <p>unicode-ibm-1264</p> <p>unicode-ibm-1265</p> <p>windows30-latin1</p> <p>windows31-latin1</p> <p>windows31-latin2</p> <p>windows31-latin5</p> <p>hp-roman8</p>

	adobe-standard-encoding ventura-us ventura-international dec-mcs pc-page-850-multilingual pc-page-852-latin2 pc8-page-437-us pc8-danish-norwegian pc-page-862-latin-hebrew pc8-turkish ibm-symbols ibm-thai hp-legal hp-pifont hp-math8 hp-ps-math hp-desktop ventura-math microsoft-publishing windows31j gb-2312 big5 cns ksc-5601 ccdc
Default	No default.
Usage Guidelines	o The document attribute default-character-set is compared to this attribute for scheduling and validation.
Attribute	Description
Name	content-orientations-supported
Input Synonym	orientations-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the page presentations supported by this logical printer.
Value Type	Multiple values.
Values	Any of the following fixed values: landscape portrait reverse-portrait reverse-landscape
Default	No value.
Usage Guidelines	o The document attribute content-orientation is compared to this attribute for validation.
Attribute	Description

Name	descriptor																				
Type	Settable with the pdcreate or pdset commands.																				
Explanation	A textual description of this logical printer.																				
Value Type	Single value.																				
Values	A text string up to 4095 characters that contains a description of this logical printer.																				
Default	No default.																				
Usage Guidelines																					
Attribute	Description																				
Name	document-attributes-supported																				
Type	Settable with the pdcreate or pdset commands.																				
Explanation	Identifies the document attributes supported by this logical printer. This attribute is checked during validation to allow jobs whose documents have these attributes to be sent to this logical printer.																				
Value Type	Multiple values.																				
Values	Any valid document attributes listed in pd_att_document(5).																				
Default	No default																				
Usage Guidelines																					
Attribute	Description																				
Name	document-formats-supported																				
Type	Settable with the pdcreate or pdset commands.																				
Explanation	Identifies the document formats supported by this logical printer.																				
Value Type	Multiple values.																				
Values	Any of the following fixed values:																				
	<table> <tr> <th>Fixed Value</th><th>Input Synonym</th></tr> <tr> <td>ascii</td><td></td></tr> <tr> <td>pcl</td><td>hppcl</td></tr> <tr> <td></td><td>hp-pcl</td></tr> <tr> <td>hpgl</td><td>hp-gl</td></tr> <tr> <td>pjl</td><td></td></tr> <tr> <td>postscript</td><td>ps</td></tr> <tr> <td>ipds</td><td></td></tr> <tr> <td>ppds</td><td></td></tr> <tr> <td>escapep</td><td></td></tr> </table>	Fixed Value	Input Synonym	ascii		pcl	hppcl		hp-pcl	hpgl	hp-gl	pjl		postscript	ps	ipds		ppds		escapep	
Fixed Value	Input Synonym																				
ascii																					
pcl	hppcl																				
	hp-pcl																				
hpgl	hp-gl																				
pjl																					
postscript	ps																				
ipds																					
ppds																					
escapep																					

	epson ddif interpress iso-6429 line-data modca regis scs spdl tek4014 pds igp codev dsc-dse wps ln03 ccitt quic cpap dec-ppl simple-text npap doc impress pinwriter npdl nec201pl automatic pages lips tiff diagnostic capsl excl lcds xes passthru
Default	No values.
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute document-format is compared to this attribute for validation.
Attribute	Description
Name	document-types-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the types of documents that are supported by this logical printer.
Value Type	Multiple values.
Values	Any of the following fixed values: file-reference printable
Default	No values.

Usage Guidelines	o The document attribute document-type is compared to this attribute for validation.							
Attribute	Description							
Name	enabled							
Type	Non Settable							
Explanation	Indicates if this logical printer is enabled and can accept print jobs.							
Value Type	Single value.							
Values	One of the following fixed values: <table><tr><td>Fixed Value</td><td>Input Synonym</td></tr><tr><td>true</td><td>yes</td></tr><tr><td>false</td><td>no</td></tr></table> Implicitly set and reset by the pdenable and pddisable commands.		Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym							
true	yes							
false	no							
Default	false							
Usage Guidelines								
Attribute	Description							
Name	end-message-supported							
Type	Settable with the pdcreate or pdset commands.							
Explanation	Indicates whether jobs submitted to this logical printer can specify an end message.							
Value Type	Single value.							
Values	One of the following fixed values: <table><tr><td>Fixed Value</td><td>Input Synonym</td></tr><tr><td>true</td><td>yes</td></tr><tr><td>false</td><td>no</td></tr></table>		Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym							
true	yes							
false	no							
Default	No value.							
Usage Guidelines	o The job attribute job-end-message is compared to this attribute for validation.							
Attribute	Description							
Name	fonts-supported							
Type	Settable with the pdcreate or pdset commands, or the printer attribute file.							
Usage								

Explanation	Identifies the fonts supported by this printer.
Value Type	Multiple values
Values	Any text string that contains the font names supported by this printer.
Default	No default.
Usage Guidelines	o The document attribute default-font is compared to this attribute for scheduling and validation.
Attribute	Description
Name	input-trays-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies the types of input trays that are supported by this logical printer.
Value Type	Multiple values.
Values	A text string of up to 255 characters that describes the input tray, or one of the following fixed values: top middle bottom envelope manual large-capacity main side
Default	No value.
Usage Guidelines	o The document attribute input-tray-select and default-input-tray are compared to this attribute for validation.
Attribute	Description
Name	interface-program-methods-supported
Type	Settable with the pdcreate, pdset commands, or printer attribute file.
Explanation	Identifies the interface program invocation methods supported by this printer.
Value Type	Single value.
Values	Any of the following fixed values: hpdps lp

Default	No default.
Usage Guidelines	o The job attribute interface-program-method is compared to this attribute for validation.
Attribute	Description
Name	job-attributes-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the job attributes supported by this printer.
Value Type	Multiple values.
Values	Any valid job attributes listed in pd_att_job(5).
Default	No value.
Usage Guidelines	
Attribute	Description
Name	job-finishings-supported
Type	Settable
Explanation	Indicates the finishing method supported by this printer.
Value Type	Single value.
Values	staple top-left-staple
Default	No default.
Usage Guidelines	To verify if a printer is capable of performing a finishing process on a print job.
Attribute	Description
Name	job-sheets-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the auxiliary-sheet-package values supported this printer.
Value Type	Multiple values.
Values	Any of the following fixed values: none job-separators job-set-start job-set-end job-set-wrap job-set-start-copies-separate

	job-set-end-copies-separate job-set-wrap-copies-separate job-copy-start job-copy-end job-copy-wrap
Default	No value.
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute job-sheets is compared to this attribute for validation. o The auxiliary-sheet-package object is not implemented. These values are used to select the auxiliary sheets.
Attribute	Description
Name	job-size-range-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Defines the range of job sizes (in bytes) the printer can accept.
Value Type	Single value.
Values	This is a complex attribute with the following components: <ul style="list-style-type: none"> lower limit upper limit
Syntax	Two integers separated by a colon; the value of the integers can be from 0 to 9223372036854775800 lower limit:upper limit For example: 0:1000000000
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute total-job-octets is compared to this attribute for validation.
Component	Description
lower limit	Explanation: Provides the lower limit of the job size range. Value Type: Single value. Values: <ul style="list-style-type: none"> o The unit is octets (bytes). o The first integer identifies the lower limit and must be less than or equal to the second integer (the upper limit). o If only the lower limit is supplied, the upper limit defaults to 9223372036854775800 Default: No value.
upper limit	Explanation: Provides the lower limit of the job size range. Value Type: Single value. Values: <ul style="list-style-type: none"> o The unit is octets (bytes). o The first integer identifies the lower limit and must be less than or equal to the second integer

	(the upper limit). o If only the upper limit is supplied, the lower limit defaults to 0. Default: No value.
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdcreate or pdset command.
Explanation	Identifies the people responsible for the maintenance and operation of this logical printer.
Value Type	Multiple values.
Values	A text string up to 255 characters per person that contains the name or user ID of the person responsible for this logical printer.
Default	No values.
Usage Guidelines	
Attribute	Description
Name	maximum-copies-supported
Type	Settable with the pdcreate or pdset
Explanation	Indicates the number of document copies, in a single print job, allowed by this logical printer.
Value Type	Single value.
Values	An integer from 1 to 2147483647.
Default	No value.
Usage Guidelines	o If this attribute value is blank, there is no limit to the number of document copies. o The document attribute copy-count is compared to this attribute for validation.
Attribute	Description
Name	media-supported
Type	Settable with the pdcreate or pdset
Explanation	Identifies the types of media supported by this logical printer.
Value Type	Multiple values.
Values	A text string up to 255 characters per item that contains

a unique name or any of the following fixed values:

```
iso-a4-white
iso-a4-colored
iso-a4-transparent
iso-a3-white
iso-a3-colored
iso-a5-white
iso-a5-colored
iso-b4-white
iso-b4-colored
iso-b5-white
iso-b5-colored
na-letter-white
na-letter-colored
na-letter-transparent
na-legal-white
na-legal-colored
iso-b5-envelope
iso-b4-envelope
iso-c4-envelope
iso-c5-envelope
designated-long-envelope
na-10x13-envelope
na-9x12-envelope
monarch-envelope
na-number-10-envelope
na-7x9-envelope
na-9x11-envelope
na-10x14-envelope
na-number-9-envelope
na-6x9-envelope
na-10x15-envelope
jis-b4-white
jis-b4-colored
jis-b5-white
jis-b5-colored
executive
folio
invoice
ledger
quarto
iso-a0-white
iso-a1-white
iso-a2-white
iso-a6-white
iso-a7-white
iso-a8-white
iso-a9-white
iso-a10-white
iso-b10-white
iso-b0-white
iso-b1-white
iso-b2-white
iso-b3-white
iso-b6-white
iso-b7-white
iso-b8-white
iso-b9-white
iso-b10-white
jis-b0-white
jis-b1-white
```

	jis-b2-white jis-b3-white jis-b6-white jis-b7-white jis-b8-white jis-b9-white jis-b10-white a b c d e
Default	No value.
Usage Guidelines	o The document attribute default-medium is compared to this attribute.
Attribute	Description
Name	message
Type	Settable with the pdcreate or pdset command.
Explanation	Information associated with this logical printer.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information about this logical printer.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	notification-profile
Type	Settable with the pdcreate or pdset command.
Explanation	Designates which persons are to be notified of specific events related to this logical printer, and how they are to be notified.
Value Type	Multiple values.
Values	This is a complex attribute, which has the following components: <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale
Syntax	-x "notification-profile={event-identifiers=value delivery-method=value event-comment='text' delivery-address=value locale=value}"

	<p>For example:</p> <pre>-x "notification-profile={event-identifiers= class-physical-printer-status delivery-method=electronic-mail event-comment='Its still going' delivery-address=jeff@ttank locale=C}"</pre>								
Component	Description								
event-identifiers	<p>Explanation: Specifies which events the person is to receive messages about.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the values listed for spooler attribute events-supported in pd_att_spooler(5).</p> <p>Default:</p> <ul style="list-style-type: none"> o object-deleted o object-cleaned 								
delivery-method	<p>Explanation: Specifies how the person is to receive event messages.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table border="0"> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>electronic-mail</td><td>e-mail, email</td></tr> <tr> <td>message</td><td></td></tr> <tr> <td>none</td><td></td></tr> </table> <p>Default: electronic-mail</p>	Fixed Value	Input Synonym	electronic-mail	e-mail, email	message		none	
Fixed Value	Input Synonym								
electronic-mail	e-mail, email								
message									
none									
event-comment	<p>Explanation: A text string that provides a comment that is appended to the event message.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that contains the comment.</p> <p>Default: No value.</p>								
delivery-address	<p>Explanation: The address of the person that is to receive event messages.</p> <p>Value Type: Single value.</p> <p>Values: The name and node of the person that is to receive messages concerning the operation of this logical printer.</p> <p>Default: The login of the user that created this logical printer with the pdcreate command.</p>								
locale	<p>Explanation: Defines the locale of the person to receive the messages. The locale is used to determine the language that the messages are transmitted in.</p> <p>Value Type: Single value.</p> <p>Values: The locale identification.</p> <p>Default: The locale of the person that created this logical printer with the pdcreate command.</p>								
Attribute	Description								
Name	numbers-up-supported								
Type	Settable with the pdcreate or pdset commands.								

Explanation	Identifies the number-up values supported by this printer. See the explanation of document attribute number-up in pd_att_document(5) for more information on number-up values.
Value Type	Multiple values.
Values	Any of the following fixed values: 0 1 2 4
Default	No default.
Usage Guidelines	o The document attribute number-up is compared to this attribute for validation.
Attribute	Description
Name	object-class
Type	Non Settable.
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute must be printer.
Default	printer
Usage Guidelines	
Attribute	Description
Name	output-bins-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the output bins supported by this printer.
Value Type	Multiple values.
Values	A text string of up to 255 characters that describes the output bin, or any of the following fixed values: top middle bottom side left right face-up

	face-down large private collator
Default	No default
Usage Guidelines	o The job attribute results-profile component output-bin is compared to this attribute for validation.
Attribute	Description
Name	page-select-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the types of page identifiers supported by this printer, either numeric or alphanumeric string, or both. Only the numeric type is currently supported. This attribute specifies the acceptable values that may be given for document attribute page-select; see pd_att_document(5) for more information.
Value Type	Multiple values.
Values	Any of the following fixed values: numeric
Default	No default
Usage Guidelines	o The document attribute page-select is compared to this attribute for validation.
Attribute	Description
Name	pd-gway-foreign-host
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the remote host associated with an HPDPS Gateway printer. Requests for the HPDPS Gateway printer will query the HPDPS on this host for information on how to contact the remote server.
Value Type	Single valued.
Value	A text string of up to 255 characters that contains the name of the remote host.
Default	No default.
Usage Guidelines	o This attribute is defined only for HPDPS Gateway printers. o This attribute names a host that runs a basicdsd server for the remote environment. If the remote

	logical printer belongs to an NFS diskless cluster, this attribute must name the cluster server host. Otherwise, if the remote printer belongs to a DCE Extended Environment then any host in the DCE cell (that is not a diskless client) may be chosen. Otherwise, the name of the standalone host that defines the remote logical printer must be chosen.
Attribute	Description
Name	pd-gway-foreign-printer
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the remote logical printer associated with an HPDPS Gateway printer. Requests for the HPDPS Gateway printer will be sent to this logical printer in the foreign environment.
Value Type	Single valued.
Value	A text string of up to 255 characters that contains the name of the remote logical printer in the foreign environment.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o This attribute is defined only for HPDPS Gateway printers. o The value is a logical printer name that must be recognized by the HPDPS on the remote host given by attribute pd-gway-foreign-host.
Attribute	Description
Name	plexes-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the plexes supported by this printer. Plex indicates whether the page images of the output document are conditioned for eventual one-sided or two-sided printing, and the relative orientation of consecutive pages.
Value Type	Multiple values.
Values	Any of the following fixed values: simplex (single sided) duplex (double sided) tumble (double sided, every other page upside-down)
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute plex is compared to this attribute for validation. o HPDPS defaults "sides=2" when plex=duplex. HPDPS does not currently perform any other conditioning when plex=duplex.

Attribute	Description
Name	printer-mopy-supported
Type	Settable
Explanation	Indicates if printer is capable of printing multiple copies by spooling a job in internal storage.
Value Type	Single boolean value.
Values	true, or false.
Default	No default.
Usage Guidelines	To verify if multiple copies can be printed by submitting a job to the printer once.
Attribute	Description
Name	printer-resolutions-supported
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the resolutions supported by this printer.
Value Type	Multiple values.
Values	<p>This is a complex attribute with the following components:</p> <ul style="list-style-type: none"> o An optional object identifier with one of the following fixed values: <ul style="list-style-type: none"> highest lowest medium o An integer from 0 to 2147483647.
Syntax	<p>[object identifier:]integer value</p> <p>For example: highest:600, or 300</p>
Component	Description
1st component	<p>Explanation: If given, the 1st component specifies an object identifier that is associated with the resolution value given in the 2nd component. This component is optional.</p> <p>If a job specifies the value of the default-printer-resolution attribute to be one of these object identifiers (highest/lowest/medium), this component gives the mapping of that object identifier to the actual resolution specified by the 2nd component.</p>
2nd component	Explanation: Actual resolutions in dots per inch supported by this printer.

Default	none
Usage Guidelines	o The document attribute default-printer-resolution is compared to this attribute for validation.
Attribute	Description
Name	printer-associated-printers
Type	Non Settable
Explanation	Identifies the physical printers associated with this logical printer.
Value Type	Multiple values.
Values	A text string up to 255 characters per physical printer that contains the ID of an associated physical printer. Implicitly set along with the queue attribute physical-printers-assigned of the queue associated with this logical printer when the physical printer registers.
Default	No value.
Usage Guidelines	
Attribute	Description
Name	printer-initial-value-document
Type	Settable with the pdcreate or pdset commands.
Explanation	Associates an initial-value-document object with this logical printer.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of the initial-value-document object.
Default	No default.
Usage Guidelines	o The associated initial-value-document object is used to supply values for attributes for documents submitted to this logical printer. These values override server defaults but are overridden by document attribute values specified on the command line.
Attribute	Description
Name	printer-initial-value-job
Type	Settable with the pdcreate or pdset commands.
Explanation	Associates an initial-value-job object with this logical printer.

Value Type	Single value.
Values	A text string up to 255 characters that contains the name of the initial-value-job object.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> The associated initial-value-job object is used to supply values for attributes of jobs submitted to this logical printer. These values override server defaults but are overridden by job attribute values specified on the command line.
Attribute	Description
Name	printer-locations
Input Synonym	locations
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the locations of physical printer devices, or the areas they service, that are allowed for jobs submitted through this logical printer.
Value Type	Multiple values.
Values	A text string up to 4095 characters per physical printer that contains the actual physical location of a physical printer device.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> The job attribute printer-locations-requested is compared to this attribute for validation.
Attribute	Description
Name	printer-model
Input Synonym	model
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the model ID of a physical printer device that can accept jobs submitted through this logical printer.
Value Type	Single value.
Values	A text string up to 4095 characters per physical printer that contains the make and model of a physical printer device.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> The job attribute printer-models-requested is compared to this attribute for validation.

Attribute	Description
Name	printer-name
Input Synonym	printer, logical-printer
Type	Non Settable
Explanation	Identifies this logical printer object.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this logical printer. Implicitly set from the argument (ServerName:PrinterName) specified by the pdcreate command used to create this logical printer.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute physical-printer-requested is compared to this attribute for validation. o The name must be unique within the DCE cell.
Attribute	Description
Name	printer-realization
Type	Specifiable with the pdcreate command.
Explanation	Indicates whether the printer has been designated as a physical or a logical printer.
Value Type	Single value.
Values	<p>One of the following values:</p> <p>logical physical</p> <p>Must be logical for printers created in a spooler server.</p>
Default	logical
Usage Guidelines	
Attribute	Description
Name	printers-ready
Type	Non Settable
Explanation	Identifies the physical printers associated with this logical printer that are ready to process a job. Implicitly set when a registered physical printer is identified as ready.
Value Type	Multiple values.

Values	A text string up to 255 characters per physical printer that contains the ID of a physical printer that is ready.						
Default	No value.						
Usage Guidelines	<ul style="list-style-type: none"> o A ready physical printer is one that is enabled and whose state is either idle, connecting-to-printer, or printing. 						
Attribute	Description						
Name	sides-supported						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Identifies whether this logical printer supports printing on one or two or both sides of the media.						
Value Type	Multiple values.						
Values	An integer of 1 or 2. To support both single-sided and double-sided printing, use both value 1 and value 2 as separate values for this multiple-valued attribute.						
Default	No value.						
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute sides is compared to this attribute for validation. 						
Attribute	Description						
Name	start-message-supported						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Indicates whether jobs submitted through this logical printer can specify a start message.						
Value Type	Single value.						
Values	<p>One of the following fixed values:</p> <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	No value.						
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute job-start-message is compared to this attribute for validation. 						

NAME

pd_att_phy_ptr - list of attributes for an HPDPS physical printer object

DESCRIPTION

The following is a list of valid attributes and values for the physical printer object class of HP Distributed Print Services.

- 1. Physical printers are used for job validation and scheduling.
- 2. The physical printer attributes xxx-supported are used for job validation. Both the xxx-ready and the xxx-supported attributes are used for job scheduling.
- 3. An error is issued if a xxx-ready attribute is updated with a value that is not currently assigned to the corresponding xxx-supported attribute.
- 4. Physical printers have different attribute sets based on the printer models.
- 5. Some of physical printer attributes can be specified in the printer attribute file for the printer. Note that xxx-ready attributes may not be described in the printer attribute file.

Attributes for a Physical Printer Object

Settable Attribute Listing	Specifiable Attribute Listing
<p>associated-queue attachment-types-supported character-sets-supported content-orientations-supported descriptor document-attributes-supported document-formats-supported document-types-supported end-message-supported fonts-supported input-trays-ready input-trays-supported interface-program-methods-supported job-attributes-supported job-batches-ready job-finishings-ready job-finishings-supported job-sheets-supported job-size-range-ready job-size-range-supported list-of-managers maximum-copies-supported media-ready media-supported message notification-profile notify-operator numbers-up-supported page-select-supported plexes-supported printer-community-name printer-tcpip-internet-address printer-tcpip-port-number printer-locations printer-mopy-ready printer-mopy-supported printer-pass-through printer-register-threshold printer-resolutions-supported printer-resource-context-</p>	<p>attachment-type device-name print-queue-name printer-realization printer-model</p>

printer	
printer-timeout-period	
printer-types	
sides-supported	
start-message-supported	

Table 1-1. Physical Printer Attributes

Attribute	Description
Name	associated-queue
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the queue that this physical printer receives jobs from for printing.
Value Type	Single value.
Values	A text string up to 255 that contains the ID of the queue. The ID cannot contain the cell name.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> The value for this attribute is the name of a queue object. The associated queue must exist and be in communication with the printer before the physical printer can be enabled.
Attribute	Description
Name	associated-server
Type	Non Settable
Explanation	Identifies the name of the supervisor in which this physical printer resides.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this physical printer. Implicitly set from the argument (ServerName:PrinterName) of the pdcreate command used to create this physical printer.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	attachment-type
Type	Specifiable with the pdcreate command.

Usage	Values vary by printer model.												
Explanation	Identifies how the physical printer device represented by this physical printer is attached to the supervisor and how the supervisor is to communicate with the physical printer device.												
Value Type	Single value.												
Values	<p>One of the following fixed values:</p> <table> <tr> <td>tcpip</td><td>supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.</td></tr> <tr> <td>tcpip-bsd</td><td>supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.</td></tr> <tr> <td>serial</td><td>supervisor will communicate with this printer via the serial interface.</td></tr> <tr> <td>parallel</td><td>supervisor will communicate with this printer via the parallel interface.</td></tr> <tr> <td>lp-spool-hp</td><td>supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.</td></tr> <tr> <td>lp-spool-bsd</td><td>supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.</td></tr> </table>	tcpip	supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.	tcpip-bsd	supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.	serial	supervisor will communicate with this printer via the serial interface.	parallel	supervisor will communicate with this printer via the parallel interface.	lp-spool-hp	supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.	lp-spool-bsd	supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.
tcpip	supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.												
tcpip-bsd	supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.												
serial	supervisor will communicate with this printer via the serial interface.												
parallel	supervisor will communicate with this printer via the parallel interface.												
lp-spool-hp	supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.												
lp-spool-bsd	supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.												
Default	No default.												
Usage Guidelines	<ul style="list-style-type: none"> Must be set with the pdcreate command at the time of creation. This attribute is validated against the printer attachment-types-supported attribute. 												
Attribute	Description												
Name	attachment-types-supported												
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.												
Usage	Values vary by printer model.												
Explanation	Identifies the attachment types supported by the physical												

	printer.												
Value Type	Multiple value.												
Values	Any of the following fixed values:												
	<table> <tr> <td>tcpip</td><td>supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.</td></tr> <tr> <td>tcpip-bsd</td><td>supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.</td></tr> <tr> <td>serial</td><td>supervisor will communicate with this printer via the serial interface.</td></tr> <tr> <td>parallel</td><td>supervisor will communicate with this printer via the parallel interface.</td></tr> <tr> <td>lp-spool-hp</td><td>supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.</td></tr> <tr> <td>lp-spool-bsd</td><td>supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.</td></tr> </table>	tcpip	supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.	tcpip-bsd	supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.	serial	supervisor will communicate with this printer via the serial interface.	parallel	supervisor will communicate with this printer via the parallel interface.	lp-spool-hp	supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.	lp-spool-bsd	supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.
tcpip	supervisor will communicate with this printer via the TCP/IP network with TCP port 910X protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.												
tcpip-bsd	supervisor will communicate with this printer via the TCP/IP network with the BSD (lpd) protocol. The printer must be a network peripheral. This protocol is supported by some HP JetDirect products.												
serial	supervisor will communicate with this printer via the serial interface.												
parallel	supervisor will communicate with this printer via the parallel interface.												
lp-spool-hp	supervisor will communicate with this printer using the lp spooler protocol for an HP-UX system. The printer must belong to an lp spooler on an HP-UX host.												
lp-spool-bsd	supervisor will communicate with this printer using the BSD (lpd) spooler protocol. The printer must belong to an lp/lpr spooler on a non-HP-UX host.												
Default	No default.												
Usage Guidelines	<ul style="list-style-type: none"> o The printer attribute attachment-type is compared to this attribute for validation. 												
Attribute	Description												
Name	cancel-individual-document-supported												
Type	Non Settable												
Explanation	Indicates whether the printer is capable of cancelling a single document within a multiple-document job.												
Value Type	Single Value												
Values	One of the following fixed values:												
	<table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no						
Fixed Value	Input Synonym												
true	yes												
false	no												

Default	false
Usage Guidelines	
Attribute	Description
Name	character-sets-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary printer model.
Explanation	Identifies the character set encodings supported by this printer.
Value Type	Multiple values.
Values	Any of the following fixed values: iso-ucs-2-level2 other ascii iso-latin1 iso-latin2 iso-latin3 iso-latin4 iso-latin-cyrillic iso-latin-arabic iso-latin-greek iso-latin-hebrew iso-latin5 iso-latin6 iso-text-communication half-width-katakana jis-encoding shift-jis euc-packed-format-japanese euc-fixed-width-japanese iso-reg4-united-kingdom iso-reg11-swedish-for-names iso-reg15-italian iso-reg17-spanish iso-reg21-german iso-reg60-danish-norwegian iso-reg69-french unicode ucs4 unicode-ascii unicode-latin1 unicode-ibm-2039 unicode-ibm-1261 unicode-ibm-1268 unicode-ibm-1276 unicode-ibm-1264 unicode-ibm-1265 windows30-latin1 windows31-latin1 windows31-latin2 windows31-latin5

	hp-roman8 adobe-standard-encoding ventura-us ventura-international dec-mcs pc-page-850-multilingual pc-page-852-latin2 pc8-page-437-us pc8-danish-norwegian pc-page-862-latin-hebrew pc8-turkish ibm-symbols ibm-thai hp-legal hp-pifont hp-math8 hp-ps-math hp-desktop ventura-math microsoft-publishing windows31j gb-2312 big5 cns ksc-5601 ccdc
Default	No default.
Usage Guidelines	o The document attribute default-character-set is compared to this attribute for scheduling and validation.
Attribute	Description
Name	checkpoint-formats-supported
Type	Non Settable
Explanation	Identifies checkpoints supported for paused jobs.
Value Type	Multiple values.
Values	Any of the following fixed values: Fixed Value dsf-results-profile
Default	dsf-resuls-profile
Usage Guidelines	o Used in rescheduling paused jobs. o HPDPS currently supports only dsf-results-profile. Job can be paused/resumed at results-set boundaries only.
Attribute	Description
Name	content-orientations-supported
Input	orientations-supported

Synonym	
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the page presentations supported by this printer.
Value Type	Multiple values.
Values	Any of the following fixed values: landscape portrait reverse-landscape reverse-portrait
Default	portrait
Usage Guidelines	o The document attribute content-orientation is compared to this attribute for scheduling and validation.
Attribute	Description
Name	descriptor
Type	Settable with the pdcreate or pdset command.
Explanation	A textual description of this physical printer.
Value Type	Single value.
Values	A text string up to 4095 characters that describes this physical printer.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	device-name
Type	Specifiable with the pdcreate command.
Usage	
Explanation	Identifies the name of the device special file associated with the communication interface for the printer.
Value Type	Single value.
Values	A text string that identifies the path name of the device associated with the physical printer.
Default	No default.

Usage	o Must be set with the pdcreate command at the time of creation if the attachment-type is serial or parallel. Ignored for other attachment-types.																																				
Attribute	Description																																				
Name	document-attributes-supported																																				
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.																																				
Explanation	Identifies the document attributes supported by this physical printer. This attribute is checked during validation to allow jobs whose documents have these attributes to be sent to this physical printer.																																				
Value Type	Multiple values.																																				
Values	Any of valid document attributes in pd_att_document(5).																																				
Default	No default.																																				
Usage Guidelines	o The document attributes are compared to this attribute for scheduling and validation.																																				
Attribute	Description																																				
Name	document-formats-supported																																				
Type	Settable with the pdcreate and pdset commands, or by the printer attribute file.																																				
Usage	Values vary by printer mode.																																				
Explanation	Identifies the document formats supported by this physical printer.																																				
Value Type	Multiple values.																																				
Values	Any of the following fixed values: <table> <tr> <th>Fixed Value</th><th>Input Synonym</th></tr> <tr> <td>ascii</td><td></td></tr> <tr> <td>pcl</td><td>hppcl</td></tr> <tr> <td></td><td>hp-pcl</td></tr> <tr> <td>hpgl</td><td>hp-gl</td></tr> <tr> <td>pjl</td><td></td></tr> <tr> <td>postscript</td><td>ps</td></tr> <tr> <td>ipds</td><td></td></tr> <tr> <td>ppds</td><td></td></tr> <tr> <td>escapep</td><td></td></tr> <tr> <td>epson</td><td></td></tr> <tr> <td>ddif</td><td></td></tr> <tr> <td>interpress</td><td></td></tr> <tr> <td>iso-6429</td><td></td></tr> <tr> <td>line-data</td><td></td></tr> <tr> <td>modca</td><td></td></tr> <tr> <td>regis</td><td></td></tr> <tr> <td>scs</td><td></td></tr> </table>	Fixed Value	Input Synonym	ascii		pcl	hppcl		hp-pcl	hpgl	hp-gl	pjl		postscript	ps	ipds		ppds		escapep		epson		ddif		interpress		iso-6429		line-data		modca		regis		scs	
Fixed Value	Input Synonym																																				
ascii																																					
pcl	hppcl																																				
	hp-pcl																																				
hpgl	hp-gl																																				
pjl																																					
postscript	ps																																				
ipds																																					
ppds																																					
escapep																																					
epson																																					
ddif																																					
interpress																																					
iso-6429																																					
line-data																																					
modca																																					
regis																																					
scs																																					

	spdl tek4014 pds igp codev dsc-dse wps ln03 ccitt quic cpap dec-ppl simple-text npap doc impress pinwriter npdl nec201pl automatic pages lips tiff diagnostic caps1 excl lcds xes passthru
Default	No default.
Usage Guidelines	o The document attribute document-format is compared to this attribute for scheduling and validation.
Attribute	Description
Name	document-types-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the types of documents supported by the physical printer.
Value Type	Multiple values.
Values	Any of the following fixed values: printable
Default	printable
Usage Guidelines	o The document attribute document-type is compared to this attribute for scheduling.
Attribute	Description
Name	enabled

Type	Non Settable						
Explanation	Indicates if the printer is enabled and can accept print jobs.						
Value Type	Single value.						
Values	One of the following fixed values: true false Value is set to false when the printer is disabled with the pddisable command.						
Default	false						
Usage Guidelines	o The queue associated with the printer must exist and be in communication with the printer before the printer can be enabled.						
Attribute	Description						
Name	end-message-supported						
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.						
Explanation	Indicates whether the physical printer supports the job attribute job-end-message.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	true						
Usage Guidelines	o The job attribute job-end-message is compared to this attribute for validation.						
Attribute	Description						
Name	fonts-supported						
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.						
Usage							
Explanation	Identifies the fonts supported by this printer.						
Value Type	Multiple values						
Values	Any text string that contains the font names supported by this printer.						

Default	No default.
Usage Guidelines	o The document attribute default-font is compared to this attribute for scheduling and validation.
Attribute	Description
Name	input-trays-ready
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the types of input trays (such as top or envelope) presently available in the physical printer device.
Value Type	Multiple values.
Values	A text string of up to 255 characters that describes the input tray, or one of the following fixed values: top middle bottom envelope manual large-capacity main side
Default	No default.
Usage Guidelines	
Attribute	Description
Name	input-trays-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the types of input trays (such as top or envelope) that are installed in the physical printer device.
Value Type	Multiple values.
Values	A text string of up to 255 characters that describes the input tray, or one of the following fixed values: top middle bottom envelope manual large-capacity main

	side
Default	No default
Usage Guidelines	
Attribute	Description
Name	interface-program-methods-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the invocation methods supported by this printer.
Value Type	Single value.
Values	Any of the following fixed values: hpdps lp
Default	No default.
Usage Guidelines	o The job attribute interface-program-method is compared to this attribute for validation.
Attribute	Description
Name	job-attributes-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the job attributes supported by this printer.
Value Type	Multiple values.
Values	Any of valid job attributes in pd_att_job(5).
Default	job-batch
Usage Guidelines	o The job attributes are compared to this attribute for scheduling and validation.
Attribute	Description
Name	job-batches-ready
Type	Settable attribute which job-batch values are acceptable to this physical printer.
Value Type	Multiple values.
Values	A text string up to 4095 characters in length,

	per value, that contains the job-batch name.
Default	No default.
Usage Guidelines	The value of the job attribute job-batch is compared with this attribute during job-scheduling
Attribute	Description
Name	job-finishings-ready
Type	Settable
Explanation	Indicates the finishing methods available on this printer.
Value Type	Single value.
Values	staple top-left-staple
Default	No default.
Usage Guidelines	The value of this attribute must be supported by job attribute finishings-supported
Attribute	Description
Name	job-finishings-supported
Type	Settable
Explanation	Indicates the finishing method supported by this printer.
Value Type	Single value.
Values	staple top-left-staple
Default	No default.
Usage Guidelines	To verify if a printer is capable of performing a finishing process on a print job.
Attribute	Description
Name	job-sheets-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the auxiliary sheets supported by this printer.
Value Type	Multiple values.
Values	Any of the following fixed values: none

	job-separators job-set-start job-set-end job-set-wrap job-set-start-copies-separate job-set-end-copies-separate job-set-wrap-copies-separate job-copy-start job-copy-end job-copy-wrap
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute job-sheets is compared to this attribute for scheduling and validation. o The auxiliary-sheet-package object is not implemented. These values are used to select the auxiliary sheets. <p>Most printer models support the following 2 values:</p> <ul style="list-style-type: none"> o none <p>No auxiliary sheets, or banner pages, are printed with the job.</p> <ul style="list-style-type: none"> o job-set-start <p>A banner page is printed at the start of each job-result-set. For most printers, this means a single banner page is printed before the job documents are printed.</p>
Attribute	Description
Name	job-size-range-ready
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Defines the range of job sizes in bytes the physical printer can currently accept and print.
Value Type	Single value.
Values	<p>This is a complex attribute with the following values:</p> <ul style="list-style-type: none"> o lower limit o upper limit <p>Two integers separated by a colon (:). Each integer can be from 0 to 9223372036854775800. The first integer is the lower limit and the second integer is the upper limit. The lower limit must be less than or equal to the upper limit. Unit value is octets (bytes).</p>
Syntax	lower limit:upper limit For example: 0:65000.
Default	0:9223372036854775800
Usage	<ul style="list-style-type: none"> o The range for this attribute value must be contained

Guidelines	<p>within the range of job-size-range-supported.</p> <ul style="list-style-type: none"> o The job attribute total-job-octets is compared to this attribute for scheduling.
Component	Description
lower limit	<p>Explanation: Lower limit value of job size.</p> <p>Value Type: Single</p> <p>Values: Unit value is octets (bytes). If only the lower limit is supplied, the upper limit defaults to 9223372036854775800.</p>
upper limit	<p>Explanation: Upper limit value of job size.</p> <p>Value Type: Single</p> <p>Values: Unit value is octets (bytes). If only the upper limit is supplied, the lower limit defaults to 0.</p>
Attribute	Description
Name	job-size-range-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Defines the range of job sizes this physical printer can accept.
Value Type	Single value.
Values	<p>This is a complex attribute with the following components:</p> <ul style="list-style-type: none"> o lower limit o upper limit <p>Two integers separated by a colon (:). Each integer can be from 0 to 9223372036854775800. The first integer is the lower limit and the second integer is the upper limit. The lower limit must be less than or equal to the upper limit. Unit value is octets (bytes).</p>
Syntax	<p>lower limit:upper limit</p> <p>For example: 0:65000</p>
Default	0:9223372036854775800
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute total-job-octets is compared to this attribute for validation.
Component	Description
lower limit	<p>Explanation: Lower limit value of job size.</p> <p>Value Type: Single</p> <p>Values: Unit value is octets (bytes). If only the lower limit is supplied, the upper limit defaults to 9223372036854775800.</p>
upper limit	<p>Explanation: Upper limit value of job size.</p> <p>Value Type: Single</p>

	Values: Unit value is octets (bytes). If only the upper limit is supplied, the lower limit defaults to 0.
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the people responsible for the maintenance and operation of this physical printer.
Value Type	Multiple values.
Values	A text string that contains the IDs of the people responsible for this physical printer.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	maximum-copies-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Indicates the number of document copies, in a single print job, allowed by this physical printer.
Value Type	Single value.
Values	An integer from 0 to 2147483647.
Default	2147483647
Usage Guidelines	<ul style="list-style-type: none"> o If the value for this attribute is "0", there is no limit to the number of document copies. o The document attribute copy-count is compared to this attribute for scheduling.
Attribute	Description
Name	maximum-printer-speed
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	
Explanation	Specifies the maximum speed of the printer in pages per minute.

Value Type	Single value.
Values	An integer from 0 to 2147483647
Default	none
Usage Guidelines	o The job attribute printer-speed-range-requested is compared to this attribute for scheduling.
Attribute	Description
Name	media-ready
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the media presently loaded in the printer.
Value Type	Multiple values.
Values	Any of the following fixed values or any text string up to 255 characters that contains the medium names. iso-a4-white iso-a4-colored iso-a4-transparent iso-a3-white iso-a3-colored iso-a5-white iso-a5-colored iso-b4-white iso-b4-colored iso-b5-white iso-b5-colored na-letter-white na-letter-colored na-letter-transparent na-legal-white na-legal-colored iso-b5-envelope iso-b4-envelope iso-c4-envelope iso-c5-envelope designated-long-envelope na-10x13-envelope na-9x12-envelope monarch-envelope na-number-10-envelope na-7x9-envelope na-9x11-envelope na-10x14-envelope na-number-9-envelope na-6x9-envelope na-10x15-envelope jis-b4-white jis-b4-colored jis-b5-white jis-b5-colored executive folio invoice ledger

	quarto iso-a0-white iso-a1-white iso-a2-white iso-a6-white iso-a7-white iso-a8-white iso-a9-white iso-a10-white iso-b0-white iso-b1-white iso-b2-white iso-b3-white iso-b6-white iso-b7-white iso-b8-white iso-b9-white iso-b10-white jis-b0-white jis-b1-white jis-b2-white jis-b3-white jis-b6-white jis-b7-white jis-b8-white jis-b9-white jis-b10-white a b c d e
Default	No default.
Usage Guidelines	
Attribute	Description
Name	media-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the types of media supported by the physical printer.
Value Type	Multiple values.
Values	Any of the following fixed values or any text string up to 255 characters that contains the medium names. iso-a4-white iso-a4-colored iso-a4-transparent iso-a3-white iso-a3-colored iso-a5-white iso-a5-colored iso-b4-white

iso-b4-colored
 iso-b5-white
 iso-b5-colored
 na-letter-white
 na-letter-colored
 na-letter-transparent
 na-legal-white
 na-legal-colored
 iso-b5-envelope
 iso-b4-envelope
 iso-c4-envelope
 iso-c5-envelope
 designated-long-envelope
 na-10x13-envelope
 na-9x12-envelope
 monarch-envelope
 na-number-10-envelope
 na-7x9-envelope
 na-9x11-envelope
 na-10x14-envelope
 na-number-9-envelope
 na-6x9-envelope
 na-10x15-envelope
 jis-b4-white
 jis-b4-colored
 jis-b5-white
 jis-b5-colored
 executive
 folio
 invoice
 ledger
 quarto
 iso-a0-white
 iso-a1-white
 iso-a2-white
 iso-a6-white
 iso-a7-white
 iso-a8-white
 iso-a9-white
 iso-a10-white
 iso-b0-white
 iso-b1-white
 iso-b2-white
 iso-b3-white
 iso-b6-white
 iso-b7-white
 iso-b8-white
 iso-b9-white
 iso-b10-white
 jis-b0-white
 jis-b1-white
 jis-b2-white
 jis-b3-white
 jis-b6-white
 jis-b7-white
 jis-b8-white
 jis-b9-white
 jis-b10-white
 a
 b
 c
 d

	e
Default	No default.
Usage Guidelines	o The document attribute default-medium is compared to this attribute.
Attribute	Description
Name	message
Type	Settable with the pdcreate or pdset commands.
Explanation	Information associated with the physical printer.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information about this physical printer.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	maximum-concurrent-jobs
Type	Non Settable.
Explanation	Identifies the number of of jobs that can be processed by this printer concurrently.
Value Type	Single Value.
Values	Any of the following fixed values: 1 Set by HPDPS.
Default	1
Usage Guidelines	
Attribute	Description
Name	multiple-documents-supported
Type	Non Settable.
Explanation	Identifies whether this printer is capable of processing and printing multiple documents per job.
Value Type	Single value.
Values	One of the following fixed values:

	false true
Default	true
Usage Guidelines	
Attribute	Description
Name	notification-profile
Type	Settable with the pdcreate or pdset commands.
Explanation	Designates which users are to be notified of specified events related to this physical printer, and how they are to be notified.
Value Type	Multiple values.
Values	<p>This is a complex attribute which has the following components:</p> <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale
Syntax	<pre>-x "notification-profile={event-identifiers=value delivery-method=value event-comment='text' delivery-address=value locale=value}"</pre> <p>For example:</p> <pre>-x "notification-profile={event-identifiers= class-physical-printer-status delivery-method=electronic-mail event-comment='Its still going' delivery-address=jeff@ttank locale=C}"</pre>
Component	Description
event-identifiers	<p>Explanation: Specifies the events for which the user is to receive messages.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the values listed for spooler attribute events-supported in pd_att_spooler(5).</p> <p>Default:</p> <ul style="list-style-type: none"> o object-deleted o object-cleaned o printer-needs-administrator o printer-needs-attention o printer-needs-operator o printer-paper-jam o printer-paper-out o printer-paper-output-problem o printer-timed-out

delivery-method	Explanation: Specifies how the user is to receive event messages. Value Type: Single value. Values: One of the following fixed values: Fixed Value Input Synonym electronic-mail e-mail, email message none Default: electronic-mail
event-comment	Explanation: Supplies textual information that is appended to the event message. Value Type: Single value. Values: A text string up to 4095 characters that contains the comment. Default: No default.
delivery-address	Explanation: The address of the person to receive the event messages. Value Type: Single value. Values: A text string that contains the user name and host, such as jeff@ttank Default: The login of the user that created this physical printer with the pdcreate command.
locale	Explanation: Provides the locale identification so the message is sent in the correct language. Value Type: Single value. Values: Locale identification. Default: The locale of the user that created this physical printer with the pdcreate command.
Attribute	Description
Name	numbers-up-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the number-up values supported by this printer. See the explanation of document attribute number-up in pd_att_document(5) for more information on number-up values.
Value Type	Multiple values.
Values	Any of the following fixed values: 0 1 2 4
Default	No default.
Usage Guidelines	<ul style="list-style-type: none">o The document attribute number-up is compared to this attribute for validation.
Attribute	Description

Name	notify-operator
Input Synonym	operators
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies people that are to receive the attribute job-message-to-operator message.
Value Type	Multiple values.
Values	This is a complex attribute, which has the following components: <ul style="list-style-type: none"> o delivery-method o delivery-address
Syntax	-x "notify-operator=dm value:da value" For example: message:op2@phyptr1
Component	Description
delivery-method	Explanation: Specifies how the person is to receive the messages. Value Type: Single value. Values: One of the following fixed values: electronic-mail message none Default: electronic-mail
delivery-address	Explanation: The address of the person to receive the messages. Value Type: Single value. Values: The user ID and host of the person. Default: The login of the person who created this physical printer with the pdcreate command.
Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is printer.
Default	printer
Usage Guidelines	
Attribute	Description
Name	output-bins-supported

Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the output bins supported by this printer.
Value Type	Multiple values.
Values	<p>A text string of up to 255 characters that describes the output bin, or any of the following fixed values:</p> <ul style="list-style-type: none"> top middle bottom side left right face-up face-down large private collator
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute results-profile component output-bin is compared to this attribute for validation.
Attribute	Description
Name	page-select-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the types of page identifiers supported by this printer, either numeric or alphanumeric string, or both. Only the numeric type is currently supported. This attribute specifies the acceptable values that may be given for document attribute page-select; see pd_att_document(5) for more information.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <ul style="list-style-type: none"> numeric
Default	No default
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute page-select is compared to this attribute for validation.
Attribute	Description

Name	plexes-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	Values vary by printer model.
Explanation	Identifies the plexes supported by this printer. Plex indicates whether the page images of the output document are conditioned for eventual one-sided or two-sided printing, and the relative orientation of consecutive pages.
Value Type	Multiple values.
Values	Any of the following fixed values: simplex (single sided) duplex (double sided) tumble (double sided, every other page upside-down)
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute plex is compared to this attribute for scheduling. o HPDPS defaults "sides=2" when plex=duplex. HPDPS does not currently perform any other conditioning when plex=duplex.
Attribute	Description
Name	printer-community-name
Type	Settable with the pdcreate or pdset commands.
Usage	Values vary by printer model.
Explanation	Identifies the password used for SNMP communication with the printer whose attachment-type is tcpip.
Value Type	Single value.
Values	A text string up to 255 that contains the physical printer's community name.
Default	internal
Usage Guidelines	<ul style="list-style-type: none"> o May be set with the pdcreate command at the time of physical printer creation if the attachment-type is tcpip. Ignored for other attachment-types.
Attribute	Description
Name	printer-mopy-ready
Type	Settable
Explanation	Indicates if printer printer's multiple copy feature is ready for use.

Value Type	Single boolean value.
Values	true, or false.
Default	No default.
Usage Guidelines	The value of this attribute must be supported by job attribute printer-mopy-supported.
Attribute	Description
Name	printer-mopy-supported
Type	Settable
Explanation	Indicates if printer is capable of printing multiple copies by spooling a job in internal storage.
Value Type	Single boolean value.
Values	true, or false.
Default	No default.
Usage Guidelines	To verify if multiple copies can be printed by submitting a job to the printer once.
Attribute	Description
Name	print-queue-name
Input Synonym	print-queue
Type	Specifiable with the pdcreate command.
Usage	
Explanation	Specifies which lp/bsd queue is associated with this physical printer.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of the lp/bsd queue.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> Must be set with the pdcreate command at the time of creation if the attachment-type is one of the following: <ul style="list-style-type: none"> tcpip-bsd lp-spool-hp lp-spool-bsd <p>Ignored for other attachment-types.</p> <p>The value identifies the name of the HP-UX lp spooler destination or BSD spooler queue associated with the</p>

	physical printer.
Attribute	Description
Name	printer-locations
Input Synonym	locations
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the locations of the physical printer devices.
Value Type	Multiple values.
Values	Any text string up to 4095 characters that contains the actual physical location of the physical printer devices.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute printer-locations-requested is compared to this attribute for scheduling.
Attribute	Description
Name	printer-model
Input Synonym	model
Type	Specifiable with the pdcreate command.
Explanation	Identifies the model ID defined by the manufacturer of the physical printer device.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the model ID.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o Must be set with the pdcreate command at the time of physical printer creation unless the attachment-type is one of the following: <ul style="list-style-type: none"> lp-spool-hp lp-spool-bsd o The job attribute printer-models-requested is compared to this attribute for scheduling.
Attribute	Description
Name	printer-name
Input Synonym	printer, physical-printer

Type	Non Settable
Explanation	Identifies the physical printer object.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this physical printer. Set by HPDPS from the argument used with the pdcreate command.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The job attribute physical-printer-requested is compared to this attribute for validation. o The name must be unique within the DCE cell.
Attribute	Description
Name	printer-needs-attention-time
Type	Non Settable
Explanation	Reports the amount of time the printer has been waiting for simple intervention, such as loading paper.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-needs-key-operator-attention-time
Type	Non Settable
Explanation	Reports the amount of time the printer has been waiting for attention by a key (or skilled) operator.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-pass-through
Input	other-options

Synonym	
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	
Explanation	Contains information to be passed to the printer interface program.
Value Type	Single value.
Values	A text string up to 255 characters that contains information for the printer interface program.
Default	No default.
Usage Guidelines	o The value of this attribute is passed as the 4th argument of the interface program.
Attribute	Description
Name	printer-realization
Type	Specifiable with the pdcreate command.
Explanation	Indicates whether the printer has been designated as a physical or a logical printer.
Value Type	Single value.
Values	One of the following fixed values: physical
Default	physical
Usage Guidelines	
Attribute	Description
Name	printer-resolutions-supported
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Explanation	Identifies the resolutions supported by the printer.
Value Type	Multiple Values
Values	This is a complex attribute with the following components: o An optional object identifier with one of the following fixed values: highest lowest medium

	o An integer from 0 to 2147483647.
Syntax	[object identifier:]integer value For example: highest:600, or 300
Component	Description
1st component	Explanation: If given, the 1st component specifies an object identifier that is associated with the resolution value given in the 2nd component. This component is optional. If a job specifies the value of the default-printer-resolution attribute to be one of these object identifiers (highest/lowest/medium), this component gives the mapping of that object identifier to the actual resolution specified by the 2nd component.
2nd component	Explanation: Actual resolutions in dots per inch supported by this printer.
Default	none
Usage Guidelines	o The document attribute default-printer-resolution is compared to this attribute for validation.
Attribute	Description
Name	printer-resource-context-printer
Type	Non Settable.
Explanation	Identifies the location of the information about the printer model.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the path name of the printer information location.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-state
Type	Non Settable
Explanation	Identifies the current state of the printer.
Value Type	Single value.
Values	One of the following fixed values: connecting-to-printer

	idle needs-attention needs-key-operator paused printing timed-out
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-tcpip-internet-address
Input Synonym	tcpip-internet-address, internet-address
Type	Settable with the pdcreate or pdset commands.
Usage	
Explanation	Sets the Internet Address parameter for TCP/IP attached printers. This is the Internet Protocol address assigned to the printer.
Value Type	Single value.
Values	<p>A text string up to 4095 characters that contains either a:</p> <p>Dotted decimal address A series of 4 integers within the range of 0 to 255, each separated by a period (.).</p> <p>For example: 15.0.65.103</p> <p>Hostname Such as hppddev4.cup.hp.com</p>
Default	No default.
Usage Guidelines	<p>o Must be set with the pdcreate command at the time of physical printer creation if the attachment-type is one of the following:</p> <p>tcpip tcpip-bsd lp-spool-hp lp-spool-bsd</p> <p>Ignored for other attachment-types.</p>
Attribute	Description
Name	printer-tcpip-port-number
Input Synonym	tcpip-port-number, port-number
Type	Settable with the pdcreate or pdset commands.

Usage	
Explanation	Sets the Port Number parameter for TCP/IP attached printers. This is the TCP/IP port number configured at the physical printer device.
Value Type	Single value.
Values	An integer from 9100 to 9102.
Default	9100
Usage Guidelines	<ul style="list-style-type: none"> May be set with the pdcreate command at the time of physical printer creation if the attachment-type is tcpip. Ignored for other attachment-types.
Attribute	Description
Name	printer-timeout-period
Input Synonym	timeout-period
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.
Usage	
Explanation	Specifies the amount of time (in seconds) the supervisor is allowed to connect to a shared network printer (physical printer device) after the physical printer receives a new job request.
Value Type	Single value.
Values	An integer from 0 to 2147483647. The unit is seconds.
Default	60 seconds
Usage Guidelines	<ul style="list-style-type: none"> While the supervisor is attempting to connect to the printer device, the physical printer is in the connecting-to-printer state, and it can still accept jobs. See the description of the maximum-concurrent-jobs attribute. If the supervisor cannot connect to the printer device within the specified time, the following happens unless the time is not 0: <ul style="list-style-type: none"> The printer-state becomes timed-out A warning message may be issued, depending on the notification profile of this physical printer. The supervisor continues to try to connect to the printer device until successful, or until the job is canceled or paused.
Attribute	Description
Name	printer-types
Type	Settable with the pdcreate or pdset commands.

Explanation	Identifies the marking technologies of the printer
Value Type	Multiple values.
Values	One of the following fixed values: other electrophotographic-led electrophotographic-laser electrophotographic-other impact-moving-head-dot-matrix-9-pin impact-moving-head-dot-matrix-24-pin impact-moving-head-dot-matrix-other impact-moving-head-fully-formed impact-band impact-other inkjet-aqueous inkjet-solid inkjet-other pen thermal-transfer thermal-sensitive thermal-diffusion thermal-other electro-erosion electro-static photographic-microfiche photographic-imagesetter photographic-other ion-deposition E-beam typesetter
Default	No default.
Usage Guidelines	o The job attribute printer-types-requested is compared to this attribute for scheduling and validation.
Attribute	Description
Name	registered-with-spooler
Type	Non Settable
Explanation	Indicates whether the physical printer and spooler have established communication.
Value Type	Single value.
Values	One of the following fixed values: Fixed Value Input Synonym true yes false no
Default	false
Usage Guidelines	

Attribute	Description						
Name	sides-supported						
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.						
Explanation	Identifies whether the printer can print on one or two sides of the media.						
Value Type	Multiple values.						
Values	An integer of 1 or 2. To support both single-sided and double-sided printing, use both value 1 and value 2 as separate values for this multiple-valued attribute.						
Default	No default						
Usage Guidelines	o The document attribute sides is compared to this attribute for scheduling.						
Attribute	Description						
Name	start-message-supported						
Type	Settable with the pdcreate or pdset commands, or by the printer attribute file.						
Explanation	Indicates whether the physical printer supports the job attribute job-start-message. If supported, the printer will be able to send the specified message to the persons identified in the notify-operator attribute.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	true						
Usage Guidelines	o The job attribute job-start-message is compared to this attribute for validation.						

NAME

pd_att_queue - list of attributes for an HPDPS queue object

DESCRIPTION

The following is a list of valid attributes and values for the queue object class of the HP Distributed Print Service.

Attributes for a Queue Object

Settable Attribute Listing	Specifiable Attribute Listing
backlog-lower-bound backlog-update-interval backlog-upper-bound descriptor list-of-managers message notify-operator notification-profile protected-attributes scheduler-ready	There are no specifiable attributes for a queue object.

Table 1-1. Queue Attributes

Attribute	Description
Name	associated-server
Type	Non Settable
Explanation	Indicates the name of the server in which this queue resides.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of the server. Implicitly set from the argument (ServerName:QueueName) specified when this queue was created with the pdcreate command.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	backlog-lower-bound
Type	Settable with the pdcreate and pdset commands.
Explanation	Identifies the limit (time to print jobs within the queue) below which the queue is not considered backlogged. A backlogged condition is reset when the queue-backlog value is less than this value, if backlog computing is enabled.
Value Type	Single value.

Values	[HH:]MM. Unit is minutes.
Syntax	backlog-lower-bound=[HH:]MM. Can be just minutes or hours and minutes. For example: -x "backlog-lower-bound=5" or -x "backlog-lower-bound=1:10"
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o The value for this attribute must be less than or equal to the value specified for the backlog-upper-bound attribute. o The value for the attribute backlogged is reset to false when the value for the queue-backlog attribute becomes less than this value if backlog computing is enabled (backlog-update-interval value not equal to 0).
Attribute	Description
Name	backlog-update-interval
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies how often the queue backlog is computed.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes. If set to 0, backlog computing is disabled.
Default	The greater value of the following: 1 minute or The value of the backlog-upper-bound attribute divided by 30.
Usage Guidelines	<ul style="list-style-type: none"> o The value for this attribute must be less than or equal to the value specified for the backlog-upper-bound attribute. o This default value is set when the queue is created. If the value for the backlog-upper-bound attribute is changed later with the pdset, the value for this attribute is not changed.
Attribute	Description
Name	backlog-upper-bound
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the limit (time to print the jobs in the queue) above which the queue is considered backlogged.

Value Type	Single value.						
Values	[HH:]MM. Unit is minutes.						
Default	No default.						
Usage Guidelines	<ul style="list-style-type: none"> o The value for this attribute must be greater than or equal to the value specified for the backlog-lower-bound attribute. o The value for the attribute backlogged is set to true when the value for the queue-backlog attribute exceeds this value if backlog computing is enabled. 						
Attribute	Description						
Name	backlogged						
Type	Not Settable						
Explanation	Identifies whether the queue is backlogged.						
Value Type	Single value.						
Values	One of the following fixed values: <table> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>true</td><td>yes</td></tr> <tr> <td>false</td><td>no</td></tr> </table>	Fixed Value	Input Synonym	true	yes	false	no
Fixed Value	Input Synonym						
true	yes						
false	no						
Default	false						
Usage Guidelines	<ul style="list-style-type: none"> o This attribute value is set to true when backlog computing is enabled (backlog-update-interval not equal to 0) and the queue-backlog value exceeds backlog-upper-bound. o This attribute value is set to false when the queue-backlog value falls below the backlog-lower-bound value. 						
Attribute	Description						
Name	descriptor						
Type	Settable with the pdcreate or pdset commands.						
Explanation	Provides a description of this queue.						
Value Type	Single value.						
Values	A text string up to 4095 characters that describes this queue. You may want to specify such things as: <ul style="list-style-type: none"> o The name of the department or the account number for the users of this queue. o Any other information that is unique to your company or organization. 						
Default	No default						
Usage Guidelines	<ul style="list-style-type: none"> o The use of this attribute is optional. However, a detailed description of this queue is helpful to users 						

	who want to determine where to submit a given job or administrators who want to determine which queue to associate with a logical printer.
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdcreate or pdset commands.
Explanation	Lists the people responsible for the maintenance and operation of this queue.
Value Type	Multiple values.
Values	A text string up to 255 characters per person that contains the name or ID of the person responsible for this queue.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	logical-printers-assigned
Type	Non Settable.
Explanation	Lists the logical printers associated with this queue.
Value Type	Multiple values.
Values	A text string up to 255 characters per printer that contains the logical printer global ID. Implicitly set to the logical printer attribute associated-queue value when the logical printer is created.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	logical-printers-ready
Type	Non Settable.
Explanation	Lists the enabled logical printers that are associated with this queue.
Value Type	Multiple values.

Values	A text string up to 255 characters that contains the ID of the physical printers (per printer) that are ready to accept jobs. Implicitly set when physical printers are enabled and have a state of either idle, connecting-to-printer, or printing. Implicitly reset when a physical printer is disabled, paused, timed-out, needs-attention, or needs-key-operator-attention.
Default	No default.
Usage Guidelines	o The list changes as logical printers are enabled or disabled.
Attribute	Description
Name	message
Type	Settable with the pdcreate or pdset commands.
Explanation	Provides a message associated with this queue.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information about this queue.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	notification-profile
Type	Settable with the pdcreate or pdset commands.
Explanation	Designates which persons are to be notified of specific events related to this queue, and how they are to be notified.
Value Type	Multiple Values
Values	This is a complex attribute, which has the following components: <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale
Syntax	<pre>-x "notification-profile={event-identifiers=events delivery-method=value event-comment='text' deliver-address=value locale=value}"</pre> <p>For example:</p> <pre>-x "notification-profile={event-identifiers=class-error delivery-method=electronic-mail</pre>

	event-comment='fix problem' delivery-address=mary@travel locale=C}"												
Component	Description												
event-identifiers	<p>Explanation: Specifies the events for which the person is to receive messages.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the event values listed for spooler attribute events-supported in pd_att_spooler.</p> <p>Default:</p> <ul style="list-style-type: none"> o object-deleted o object-cleaned o queue-backlogged 												
delivery-method	<p>Explanation: Specifies how the user is to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table border="0"> <tr> <td>Fixed Value</td><td>Input Synonym</td></tr> <tr> <td>electronic-mail</td><td>e-mail, email</td></tr> <tr> <td>file</td><td></td></tr> <tr> <td>file-add-to</td><td></td></tr> <tr> <td>message</td><td></td></tr> <tr> <td>none</td><td></td></tr> </table> <p>Default: electronic-mail</p>	Fixed Value	Input Synonym	electronic-mail	e-mail, email	file		file-add-to		message		none	
Fixed Value	Input Synonym												
electronic-mail	e-mail, email												
file													
file-add-to													
message													
none													
event-comment	<p>Explanation: Supplies textual information that is appended to the message</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that supplies additional information concerning the event.</p> <p>Default: No value.</p>												
delivery-address	<p>Explanation: The address of the person that is to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the name and node of the person that is to receive notification.</p> <p>Default: The login of the person that created the queue with the pdcreate command.</p>												
locale	<p>Explanation: Identifies the locale that the messages are to be sent in.</p> <p>Value Type: Single value.</p> <p>Values: A text string that identifies the locale to be used. This is used to determine the language and coded character set that the information is to be sent in.</p> <p>Default: The locale of the person that created this queue with the pdcreate command.</p>												
Attribute	Description												
Name	notify-operator												
Input Synonym	operators												
Type	Settable with the pdcreate or pdset commands.												

Explanation	Identifies people that are to receive the attribute job-message-to-operator message.
Value Type	Multiple values.
Values	This is a complex attribute with the following components: o delivery-method o delivery-address
Syntax	-x "notify-operator=delivery-method value: delivery-address value" For example: -x "notify-operator=message:op3@fastpr"
Component	Description
delivery-method	Explanation: Specifies how the person is to receive the messages. Value Type: Single Value Values: One of the following fixed values: electronic-mail message none Default: electronic-mail
delivery-address	Explanation: The address of the person that is to receive the message. Value Type: Single Value Values: A text string that contains the user ID and node of the person that is to receive the message. Default: The login of the person that created this queue.
Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is queue
Default	queue
Usage Guidelines	
Attribute	Description
Name	physical-printers-assigned
Type	Non Settable
Explanation	Lists the physical printers that receive jobs from this queue.

Value Type	Multiple values.
Values	A text string up to 255 characters per printer that contains the ID of a physical printer that is associated with this queue. Implicitly set when a physical printer registers and its associated-queue attribute value is the name of this queue.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	physical-printers-ready
Type	Non Settable
Explanation	Lists the enabled physical printers that can receive a job from this queue.
Value Type	Multiple values.
Values	A text string up to 255 characters per printer that contains the global ID of a physical printer assigned to this queue that is ready.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o A physical printer becomes ready when it is enabled and the state is either idle, connecting-to-printer, or printing.
Attribute	Description
Name	protected-attributes
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies one or more queue attributes that you do not want HPDPS operators to set or change.
Value Type	Multiple values.
Values	One or more queue attribute names.
Default	protected-attributes
Usage Guidelines	<ul style="list-style-type: none"> o Normally, anyone with DCE write permission for queues can set values for queue attributes. By default, the pd_admin and pd_operator DCE groups both have write permission. Once you define a queue attribute as a protected attribute, DCE delete permission is required to modify the attribute. Members of the pd_operator DCE group do not have delete permission unless modifications have been made to the default permissions for that group.

Attribute	Description
Name	queue-backlog
Input Synonym	backlog, current-backlog
Type	Non Settable
Explanation	Specifies the amount of time that this queue might be backlogged. This is a computed estimate of time it will take to print all of the jobs currently in the queue.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	queue-name
Type	Non Settable
Explanation	Uniquely identifies this queue object.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this queue. Set by the argument (ServerName:QueueName) of the pdcreate command when this queue was created.
Default	No default.
Usage Guidelines	o Name of this queue must be unique within the DCE cell.
Attribute	Description
Name	queue-state
Type	Non Settable
Explanation	Identifies the current state of the queue.
Value Type	Single value.
Values	One of the following fixed values: paused ready
Default	No default.
Usage	

Guidelines	
Attribute	Description
Name	scheduler-ready
Input Synonym	scheduler
Type	Settable with the pdcreate or pdset commands.
Explanation	Identifies the scheduling method used to determine the order in which jobs in the queue are sent to physical printers.
Value Type	Single Value
Values	One of the following fixed values: fifo priority-fifo Must be one of the spooler attribute schedulers-supported values.
Default	priority-fifo
Usage Guidelines	<ul style="list-style-type: none"> o This attribute can be changed even if there are jobs currently in the queue. The jobs currently in the queue will be reordered according to the new scheduler.

NAME

pd_att_spooler - list of attributes for an HPDPS spooler object

DESCRIPTION

The following is a list of valid attributes and values for the spooler object class of the HP Distributed Print Service.

Attributes for a Spooler Object

Settable Attribute Listing	Specifiable Attribute Listing
descriptor job-submission-timer list-of-managers message notification-profile printer-register-threshold protected-attributes security-level	There are no specifiable attributes for a spooler object.

Table 1-1. Spooler Attributes

Attribute	Description
Name	cancel-individual-document-supported
Type	Non Settable
Explanation	Indicates whether the spooler is capable of canceling individual documents within a multiple-document job.
Value Type	Single value.
Values	The acceptable value is false or its synonym.
Default	false
Usage Guidelines	
Attribute	Description
Name	descriptor
Type	Settable with the pdset command.
Explanation	Provides a description of this spooler.
Value Type	Single value.
Values	A text string up to 4095 characters that contains a description of this spooler.
Default	No default.
Usage Guidelines	

Attribute	Description
Name	document-attributes-supported
Type	Non Settable
Explanation	Identifies the document attributes supported by the spooler.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <p>content-orientation See Note at end of list</p> <p>copy-count</p> <p>default-input-tray See Note at end of list</p> <p>default-medium See Note at end of list</p> <p>document-comment</p> <p>document-content</p> <p>document-file-name</p> <p>document-format</p> <p>document-sequence-number</p> <p>document-type</p> <p>initial-value-document</p> <p>object-class</p> <p>octet-count</p> <p>page-count</p> <p>plex See Note at end of list</p> <p>print-quality See Note at end of list</p> <p>printer-initial-value-document</p> <p>printer-pass-through See Note at end of list</p> <p>sides See Note at end of list</p> <p>transfer-method</p> <p>Note: This value is dynamic; it is added and removed depending on the default values of the physical printers that are registered.</p>
Default	No default.
Usage Guidelines	
Attribute	Description
Name	events-supported
Type	Non Settable
Explanation	Lists the events supported by the spooler.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <p>class-aborted</p> <p>class-error</p> <p>class-job-attention</p> <p>class-job-default</p> <p>class-job-problem</p>

```

class-job-status
class-logical-printer-attention
class-logical-printer-default
class-logical-printer-configuration
class-logical-printer-status
class-queue-attention
class-queue-default
class-queue-configuration
class-queue-status
class-server-attention
class-server-default
class-server-configuration
class-server-status
class-report
class-state-changed
class-warning
close-to-discard-time
internal-server-error
job-aborted-by-server
job-assigned-to-queue
job-canceled-by-operator
job-canceled-by-user
job-cannot-be-scheduled
job-completed
job-discarded
job-modified
job-paused
job-promoted
job-requeued
job-resubmitted
job-resumed
job-state-changed
job-submission-not-complete
job-timed-out
no-document
object-cleaned
object-created
object-deleted
object-modified
object-paused
object-resumed
past-discard-time
printer-disabled
printer-enabled
printer-registered
printer-shutdown-job-requeued
printer-unregistered
processing-started
queue-backlogged
queue-no-longer-backlogged
queue-state-changed
server-startup-complete
server-shutdown-complete
server-shutdown-started
server-state-changed

```

Default

No default.

Usage
Guidelines

Attribute	Description
Name	job-attributes-supported
Type	Non Settable
Explanation	Identifies the job attributes supported by this spooler.
Value Type	Multiple values.
Values	Any of the following fixed values: completion-time current-job-state initial-value-job intervening-jobs job-batch job-client-id job-comment job-copies-completed job-discard-time job-end-message job-finishing job-hold job-identifier job-message-from-administrator job-message-to-operator job-name job-originator job-owner job-print-after job-priority job-retention-period job-start-message job-state-reasons job-submission-complete modification-time name-of-last-accesser new-job-identifier notification-profile number-of-documents object-class octets-completed on-request-resources-required pages-completed physical-printer-requested previous-job-state print-checkpoint printer-initial-value-job printer-locations-requested printer-models-requested printer-name-requested printers-assigned printers-used processing-time queue-assigned results-profile started-printing-time submission-time total-job-octets user-locale user-name

Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-state-reasons-supported
Type	Non Settable
Explanation	Identifies the job-state reasons that are supported by this spooler.
Value Type	Multiple values.
Values	Any of the following fixed values: aborted-by-system canceled-by-operator canceled-by-user successful-completion job-hold-set job-print-after-specified required-resources-not-ready required-resources-not-supported
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-states-supported
Type	Non Settable
Explanation	Identifies the job-states supported by this spooler.
Value Type	Multiple values.
Values	Any of the following fixed values: canceled held paused pending pre-processing processing retained terminating timed-out unknown
Default	No default.
Usage Guidelines	

Attribute	Description
Name	job-submission-timer
Type	Settable with the pdset command.
Explanation	Specifies the maximum time to wait between job submissions.
Value Type	Single value.
Values	[HH:]MM. Unit is minutes.
Default	30 minutes
Usage Guidelines	
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdset command.
Explanation	Lists the people that are responsible for the maintenance and operation of this spooler.
Value Type	Multiple values.
Values	A text string up to 255 characters that contain the name or user IDs of the people responsible for this spooler.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	locale
Type	Non Settable
Explanation	Identifies the locale for this spooler.
Value Type	Single value.
Values	A text string up to 255 characters that contains the locale ID of this spooler.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> Derived from the environment variables for this spooler at startup time. The priority sequence to obtain the locale from the environment variables is:

	1. LC_ALL 2. LC_MESSAGES 3. LANG
Attribute	Description
Name	logical-printers-ready
Type	Non Settable
Explanation	Identifies which logical printers within this spooler are ready to accept jobs.
Value Type	Multiple values.
Values	A text string up to 255 characters that contains the ID of the enabled logical printers. Implicitly set and reset when logical printers are enabled or disabled within this spooler.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	logical-printers-supported
Type	Non Settable
Explanation	Identifies the logical printers that are supported by this spooler.
Value Type	Multiple values.
Values	Any text string up to 255 characters per logical printer. Implicitly set when logical printers are created or deleted within this spooler.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	message
Type	Settable with the pdset command.
Explanation	Contains information associated with this spooler.
Value Type	Single value.
Values	A text string up to 4095 characters that contains information that is associated with this spooler.
Default	No default.

Usage Guidelines	
Attribute	Description
Name	modify-individual-document-supported
Type	Non Settable
Explanation	Indicates whether this spooler is capable of modifying individual documents within a multiple document job.
Value Type	Single value.
Values	The only value is false.
Default	false
Usage Guidelines	
Attribute	Description
Name	multiple-documents-supported
Type	Non Settable
Explanation	Indicates whether multiple documents per job are supported by this spooler.
Value Type	Single Value
Values	The only value is true.
Default	true
Usage Guidelines	
Attribute	Description
Name	notification-delivery-methods-supported
Type	Non Settable
Explanation	Identifies the methods that are supported by this spooler for sending messages to the person that is to receive them.
Value Type	Multiple values.
Values	Any of the following fixed values: electronic-mail message file file-add-to

	none										
Default	No default.										
Usage Guidelines											
Attribute	Description										
Name	notification-profile										
Type	Settable with the pdset command.										
Explanation	Designates which users are to be notified of specific events related to this spooler, and how they are to be notified.										
Value Type	Multiple Values										
Values	<p>This is a complex attribute, which has the following components:</p> <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale 										
Syntax	<pre>-x "notification-profile={event-identifiers=events delivery-method=value event-comment='text' delivery-address=value locale=value}"</pre> <p>For example:</p> <pre>-x "notification-profile={event-identifiers= class-server-status delivery-method=electronic-mail event-comment='Too much data' delivery-address=Tom@master locale=C}"</pre>										
Component	Description										
event-identifiers	<p>Explanation: Specifies the events for which the person is to receive messages.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the values listed for spooler attribute events-supported.</p> <p>Default:</p> <ul style="list-style-type: none"> o internal-server-error o object-cleaned o object-deleted o server-shutdown-complete 										
delivery-method	<p>Explanation: Specifies how the user is to the receive event messages.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table border="0"> <tr> <td>Fixed Value</td> <td>Input Synonym</td> </tr> <tr> <td>electronic-mail</td> <td></td> </tr> <tr> <td>message</td> <td>e-mail, email</td> </tr> <tr> <td>file</td> <td></td> </tr> <tr> <td>file-add-to</td> <td></td> </tr> </table>	Fixed Value	Input Synonym	electronic-mail		message	e-mail, email	file		file-add-to	
Fixed Value	Input Synonym										
electronic-mail											
message	e-mail, email										
file											
file-add-to											

	<p>none</p> <p>Default: message</p>
event-comment	<p>Explanation: A text string of information that is appended to the event message.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that contains the comment.</p> <p>Default: No value.</p>
delivery-address	<p>Explanation: The address of the person that is to receive event messages concerning this spooler.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the user ID and node of the person that is to receive the event messages.</p> <p>Default: The login of the user that created this spooler.</p>
locale	<p>Explanation: Identifies the locale so the messages can be sent in the language of the person that is to receive them.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the locale of the person that is to receive the messages.</p> <p>Default: The locale of the person that created this spooler.</p>
Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is server.
Default	server
Usage Guidelines	
Attribute	Description
Name	object-classes-supported
Type	Non Settable
Explanation	Lists the object classes supported by this spooler.
Value Type	Multiple values.
Values	<p>Any of the following fixed values:</p> <p>document</p> <p>initial-value-document</p>

	initial-value-job job log printer queue server
Default	No default.
Usage Guidelines	
Attribute	Description
Name	physical-printers-ready
Type	Non Settable
Explanation	Identifies the supported physical printers that are ready to receive jobs from queues in this spooler.
Value Type	Multiple values.
Values	A text string up to 255 characters that contains the ID of the physical printers (per printer) that are ready to accept jobs. Implicitly set when physical printers associated with queues in this spooler are enabled and have a state of either idle, connecting-to-printer, or printing. Implicitly reset when a physical printer is disabled, paused, timed-out, needs-attention, or needs-key-operator-attention.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> o A physical printer is ready when it is enabled and the state is either idle, connecting-to-printer, or printing.
Attribute	Description
Name	physical-printers-supported
Type	Non Settable
Explanation	Identifies the physical printers that are supported by this spooler.
Value Type	Multiple values.
Values	A text string up to 255 characters per physical printer that contains the IDs of the physical printers that are associated with the queues contained in this spooler. Implicitly set when physical printers associated with queues in this spooler register with this spooler.
Default	No default.
Usage Guidelines	

Attribute	Description
Name	printer-register-threshold
Input Synonym	register-threshold
Type	Settable with the pdset command.
Explanation	Indicates the amount of time this spooler waits after being re-initialized for a physical printer to re-register. When the physical printer re-registers, it provides information about jobs that were processing when this spooler was shutdown.
Value Type	Single value.
Values	[HH:]MM
Default	10 minutes
Usage Guidelines	<ul style="list-style-type: none"> o If the physical printer does not re-register within the time period and provide status about the jobs it received, the spooler changes the state of the jobs from unknown to timed-out.
Attribute	Description
Name	problem-child
Type	Non Settable
Explanation	Identifies whether one of the objects managed by this spooler has a problem or not.
Value Type	Single value.
Values	One of the following fixed values: true false
Default	No default.
Usage Guidelines	Used by SAM to determine the problem status of this spooler.
Attribute	Description
Name	protected-attributes
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies one or more spooler attributes that you do not want HPDPS operators to set or change.
Value Type	Multiple values.
Values	One or more spooler attribute names.

Default	protected-attributes
Usage Guidelines	<ul style="list-style-type: none"> Normally, anyone with DCE write permission for spooler can set values for spooler attributes. By default, the pd_admin and pd_operator DCE groups both have write permission. Once you define a spooler attribute as a protected attribute, DCE delete permission is required to modify the attribute. Members of the pd_operator DCE group do not have delete permission unless modifications have been made to the default permissions for that group.
Attribute	Description
Name	queues-supported
Type	Non Settable
Explanation	Identifies the queues that are contained in this spooler.
Value Type	Multiple values.
Values	A text string up to 255 characters that contains the IDs of the queues contained in this spooler. Implicitly set when queues are created or deleted.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	schedulers-supported
Type	Non Settable
Explanation	Identifies the scheduling algorithms supported by this spooler.
Value Type	Multiple values.
Values	Any of the following fixed values:
	fifo priority-fifo
Default	fifo priority-fifo
Usage Guidelines	
Attribute	Description
Name	security-level

Type	Settable with the pdset command.
Explanation	Identifies the security level for this spooler.
Value Type	Multiple values.
Values	The only value supported is medium.
Default	medium
Usage Guidelines	
Attribute	Description
Name	server-hostname
Input Synonym	hostname
Type	Non Settable
Explanation	Name of the host processor on which this spooler is running.
Value Type	Single Value
Values	A text string up to 4095 characters that contains the hostname of this server. Implicitly set to the hostname where this spooler was created.
Syntax	node.node.node For example: boxer.denver.gym.com
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-ip-address
Input Synonym	ip-address, i-p-address
Type	Non Settable
Explanation	The Internet Address of the host processor on which this spooler is running.
Value Type	Single Value
Values	A text string that contains the Internet Address for this spooler in format of 4 integers in series within the range of 0 to 255. Each integer is separated from the others by a period (.). Implicitly set to the ip-address of the host on which this spooler is created.

Syntax	nn.nn.nn.nn For example: 15.1.1.1
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-name
Type	Non Settable
Explanation	Uniquely identifies this spooler.
Value Type	Single value.
Values	A text string up to 255 characters that contains name of this spooler. Implicitly set from the name specified when the startspl utility is run.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-state
Type	Non Settable
Explanation	Identifies the current state of this spooler.
Value Type	Single value.
Values	One of the following fixed values: initializing ready terminating unavailable
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-type
Type	Non Settable
Explanation	Identifies the type of server, either spooler or supervisor.

Value Type	Single value.
Values	The value is spooler.
Default	spooler
Usage Guidelines	
Attribute	Description
Name	transfer-methods-supported
Type	Non Settable
Explanation	Identifies the transfer-methods supported by this spooler.
Value Type	Multiple values.
Values	Any of the following fixed values: dce-pipe-pull with-request
Default	dce-pipe-pull with-request
Usage Guidelines	<ul style="list-style-type: none"> o The document attribute transfer-method is compared to this attribute for validation and scheduling.

NAME

pd_att_supervisor - list of attributes for an HPDPS supervisor object

DESCRIPTION

The following is a list of valid attributes and values for the supervisor object class of the HP Distributed Print Service.

Attributes for a Supervisor Object

Settable Attribute Listing	Specifiable attribute Listing
descriptor job-submission-timer list-of-managers message notification-profile protected-attributes security-level server-resource-context-printer	There are no specifiable attributes for a supervisor server object.

Table 1-1. Supervisor Attributes

Attribute	Description
Name	cancel-individual-document-supported
Type	Non Settable
Explanation	Indicates whether this supervisor is capable of cancelling individual documents within a multiple document job.
Value Type	Single value.
Values	The only value is false.
Default	false
Usage Guidelines	
Attribute	Description
Name	descriptor
Type	Settable with the pdset command.
Explanation	Provides a description of this supervisor.
Value Type	Single value.
Values	A text string up to 4095 characters that describes this supervisor.
Default	No default.
Usage Guidelines	

Attribute	Description
Name	events-supported
Type	Non Settable
Explanation	Lists the events supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values: checkpoint-taken class-aborted class-error class-physical-printer-attention class-physical-printer-default class-physical-printer-configuration class-physical-printer-status class-report class-server-attention class-server-default class-server-configuration class-server-status class-state-changed class-warning detailed-messages document-aborted-by-printer document-aborted-by-server document-cancelled-at-printer document-content file-transferred internal-server-error job-aborted-by-server job-cancelled-by-operator job-cancelled-by-user job-completed no-document no-resource object-cleaned object-created object-deleted object-modified object-paused object-resumed other-error other-warning printer-disabled printer-enabled printer-function-unavailable printer-needs-administrator printer-needs-attention printer-needs-operator printer-paper-jam printer-paper-out printer-paper-output-problem printer-registered printer-shutdown-job-requeued printer-state-changed printer-timed-out printer-toner-low

	printer-unregistered processing-started resource-needs-attention resource-needs-operator server-shutdown-complete server-shutdown-started server-startup-complete server-state-changed unable-to-register unrecognized-resource
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-state-reasons-supported
Type	Non Settable
Explanation	Identifies the job-state reasons that are supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values: successful-completion cancelled-by-user cancelled-by-operator aborted-by-system
Default	No default.
Usage Guidelines	
Attribute	Description
Name	job-states-supported
Type	Non Settable
Explanation	Identifies the job-states supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values: cancelled processing terminating
Default	
Usage Guidelines	

Attribute	Description
Name	job-submission-timer
Type	Settable with the pdset command.
Explanation	Specifies the maximum time to wait for job submissions to complete.
Value Type	Single value.
Values	[HH:]MM. The unit is minutes.
Default	30 minutes
Usage Guidelines	
Attribute	Description
Name	list-of-managers
Input Synonym	managers
Type	Settable with the pdset command.
Explanation	Lists the people that are responsible for the maintenance and operation of this supervisor.
Value Type	Multiple values.
Values	A text string that contains the name or user IDs of the people that are responsible for this supervisor.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	locale
Type	Non Settable
Explanation	Identifies the locale of this supervisor.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the locale ID of this supervisor.
Default	No default.
Usage Guidelines	<ul style="list-style-type: none"> Derived from the environment variable for this supervisor at the startup time. The priority sequence to obtain the locale from the environment variables is:

	1. LC_ALL 2. LC_MESSAGES 3. LANG
Attribute	Description
Name	message
Type	Settable with the pdset command.
Explanation	Information associated with this supervisor.
Value Type	Single value.
Values	A text string up to 4095 characters that contains the information associated with this supervisor.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	multiple-documents-supported
Type	Non Settable
Explanation	Indicates whether multiple documents per job are supported by the supervisor.
Value Type	Single value.
Values	The only value is true.
Default	true
Usage Guidelines	
Attribute	Description
Name	notification-delivery-methods-supported
Type	Non Settable
Explanation	Identifies the notification delivery methods supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values:
	electronic-mail message file file-add-to none

Default	No default.												
Usage Guidelines													
Attribute	Description												
Name	notification-profile												
Type	Settable with the pdset command.												
Explanation	Designates which users are to be notified of specific events related to the supervisor, and how they are to be notified.												
Value Type	Multiple values.												
Values	<p>This is a complex attribute, which has the following components:</p> <ul style="list-style-type: none"> o event-identifiers o delivery-method o event-comment o delivery-address o locale 												
Syntax	<pre>-x "notification-profile={event-identifiers=events delivery-method=value event-comment='text' delivery-address=value locale=value}"</pre> <p>For example:</p> <pre>-x "notification-profile={event-identifiers= object-cleaned object-deleted delivery-method=electronic-mail event-comment='Better check' delivery-address=dave@final local=C}"</pre>												
Component	Description												
event-identifiers	<p>Explanation: Specifies the events for which the person is to receive messages.</p> <p>Value Type: Multiple values.</p> <p>Values: Any of the values listed for supervisor attribute events-supported.</p> <p>Default:</p> <ul style="list-style-type: none"> o internal-server-error o object-cleaned o object-deleted o server-shutdown-complete 												
delivery-method	<p>Explanation: Specifies how the user is to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: One of the following fixed values:</p> <table border="0"> <tr> <td>Fixed Value</td> <td>Input Synonym</td> </tr> <tr> <td>electronic-mail</td> <td>e-mail, email</td> </tr> <tr> <td>message</td> <td></td> </tr> <tr> <td>file</td> <td></td> </tr> <tr> <td>file-add-to</td> <td></td> </tr> <tr> <td>none</td> <td></td> </tr> </table>	Fixed Value	Input Synonym	electronic-mail	e-mail, email	message		file		file-add-to		none	
Fixed Value	Input Synonym												
electronic-mail	e-mail, email												
message													
file													
file-add-to													
none													

	Default: electronic-mail
event-comment	<p>Explanation: Information that is appended to the event message.</p> <p>Value Type: Single value.</p> <p>Values: A text string up to 4095 characters that contains the comment.</p> <p>Default: No default.</p>
delivery-address	<p>Explanation: The address of the person that is to receive the event messages.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the user ID and node of the person that is to receive the event messages.</p> <p>Default: The login of the person that created this supervisor.</p>
locale	<p>Explanation: Identifies the locale of the person to receive the event messages. The locale is used to determine which language the messages are to be sent in.</p> <p>Value Type: Single value.</p> <p>Values: A text string that contains the locale of the person that is to receive the messages.</p> <p>Default: The locale of the person that created this supervisor.</p>
Attribute	Description
Name	object-class
Type	Non Settable
Explanation	Identifies the object class to which this object belongs.
Value Type	Single value.
Values	The value of this attribute is server.
Default	server
Usage Guidelines	
Attribute	Description
Name	object-classes-supported
Type	Non Settable
Explanation	Lists the object classes supported by this server.
Value Type	Multiple values.
Values	Any of the following fixed values:
	<p>job</p> <p>document</p>

	printer server log
Default	No default.
Usage Guidelines	
Attribute	Description
Name	physical-printers-ready
Type	Non Settable.
Explanation	Identifies the physical printers contained within this supervisor that are ready to accept jobs.
Value Type	Multiple values.
Values	A text string up to 255 characters that contains the ID of the physical printers (per printer) that are ready to accept jobs. Implicitly set when physical printers are enabled and have a state of either idle, connecting-to-printer, or printing. Implicitly reset when a physical printer is disabled.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	physical-printers-supported
Type	Non Settable
Explanation	Identifies the physical printers that are contained in this supervisor.
Value Type	Multiple values.
Values	Any text string up to 255 characters that contains the IDs of the physical printers that are contained within this supervisor. Implicitly set and reset when physical printers are created or deleted.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-models-supported
Type	Non Settable

Explanation	Identifies the physical printer models that can be created within this supervisor.
Value Type	Multiple values.
Values	Any text string up to 255 characters that contains the IDs of the physical printer models that can be created within this supervisor.
Default	No default.
Usage Guidelines	
Attribute	Description
Name	printer-states-supported
Type	Non Settable
Explanation	Identifies the printer states that are supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values: connecting-to-printer idle needs-attention needs-key-operator paused printing timed-out
Default	No default.
Usage Guidelines	
Attribute	Description
Name	problem-child
Type	Non Settable
Explanation	Identifies if one of the objects managed by this supervisor has a problem or not.
Value Type	Single values.
Values	Any of the following fixed values: true false
Default	No default.
Usage Guidelines	

Attribute	Description
Name	protected-attributes
Type	Settable with the pdcreate or pdset commands.
Explanation	Specifies one or more supervisor attributes that you do not want HPDPS operators to set or change.
Value Type	Multiple values.
Values	One or more supervisor attribute names.
Default	protected-attributes
Usage Guidelines	<ul style="list-style-type: none"> Normally, anyone with DCE write permission for supervisor can set values for supervisor attributes. By default, the pd_admin and pd_operator DCE groups both have write permission. Once you define a supervisor attribute as a protected attribute, DCE delete permission is required to modify the attribute. Members of the pd_operator DCE group do not have delete permission unless modifications have been made to the default permissions for that group.
Attribute	Description
Name	security-level
Type	Settable with the pdset command.
Explanation	Identifies the security level for this supervisor.
Value Type	Single value.
Values	<p>The following value is the only supported value.</p> <p>medium</p>
Default	medium.
Usage Guidelines	
Attribute	Description
Name	server-hostname
Input Synonym	hostname
Type	Non Settable
Explanation	Name of the host processor on which this supervisor is running.
Value Type	Single value.

Values	A text string up to 255 characters that contains the name of the host on which this supervisor is running. Implicitly set to the name of the host on which this supervisor was created.
Syntax	node.node.node For example: paper.endicott.jc.com
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-ip-address
Input Synonym	ip-address, i-p-address
Type	Non Settable
Explanation	The Internet Address of the host processor on which this supervisor is running.
Value Type	Single value.
Values	A text string that contains the Internet Address for this supervisor. Implicitly set to the ip-address of the host on which this supervisor was created.
Syntax	nn.nn.nn.nn Each integer can be from 0 to 255. For example: 15.1.1.1
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-name
Type	Non Settable
Explanation	Uniquely identifies this supervisor.
Value Type	Single value.
Values	A text string up to 255 characters that contains the name of this supervisor. Implicitly set from the name specified by the startsuv utility when this supervisor was created.
Default	No default.
Usage	

Guidelines	
Attribute	Description
Name	server-resource-context-printer
Type	Settable with the pdset command.
Explanation	Specifies the pathnames where printer model configuration information reside.
Value Type	Single Value
Values	A text string up to 4095 characters that contains the pathname(s).
Default	/opt/pd/lib/model
Usage Guidelines	<ul style="list-style-type: none"> o The supervisor uses the value of this attribute to locate printer model information when creating a physical printer object. o Multiple paths are separated by a colon (":"). For example, "/opt/pd/lib/model:/home/user/model". The supervisor looks for printer model information beneath each path in the order specified.
Attribute	Description
Name	server-state
Type	Non Settable
Explanation	Identifies the current state of this supervisor.
Value Type	Single value.
Values	One of the following fixed values: initializing ready terminating unavailable
Default	No default.
Usage Guidelines	
Attribute	Description
Name	server-type
Type	Non Settable
Explanation	Identifies the type of server, either spooler or supervisor.
Value Type	Single value.

Values	The value is supervisor
Default	supervisor
Usage Guidelines	
Attribute	Description
Name	transfer-methods-supported
Type	Non Settable
Explanation	Identifies the transfer-methods supported by this supervisor.
Value Type	Multiple values.
Values	Any of the following fixed values: dce-pipe-pull with-request
Default	dce-pipe-pull with-request
Usage Guidelines	o The document attribute transfer-method is compared to this attribute for validation and scheduling.

NAME

pfs_exports, pfs_xtab - directories to export to PFS clients

SYNOPSIS

```
/etc/pfs_exports
/etc/pfs_xtab
```

DESCRIPTION

The `/etc/pfs_exports` file contains entries for directories that can be exported to PFS clients. This file is read automatically by the `pfs_exportfs(1M)` command. If you change this file, you must run `pfs_exportfs(1M)` for the changes to affect the daemon's operation.

It is recommended that you place a command in `rc(1M)` so that `pfs_exportfs(1M)` is executed at boot time.

The `/etc/pfs_xtab` file contains entries for directories that are *currently* exported. You should not modify this file directly. Future updates to the **PFS** package will contain libraries for editing this file. (Use the `-u` option of `pfs_exportfs` to remove entries from this file).

An entry for a directory consists of a line of the following form:

```
directory -option [, option] ...
```

where

directory is the pathname of a directory (or file).

option is one of

```
access=client [ : client ] . . .
```

Give mount access to each *client* listed. A *client* must be a hostname. The default value allows any machine to mount the given directory.

A '#' (pound-sign) anywhere in the file indicates a comment that extends to the end of the line.

EXAMPLE

```
/usr/local    # export to the world
/usr2         -access=hermes:zip:tutorial  # export to only these machines
```

AUTHOR

`pfs_exports` was developed by Young Minds, Inc.

FILES

```
/etc/pfs_exports
/etc/pfs_xtab
/etc/hosts
```

SEE ALSO

`hosts(5)`, `pfs_exportfs(1M)`, `pfsd(1M)`, `pfs_mountd(1M)`, `rc(1M)`.

NAME

pfs_fstab, mtab - static file system mounting table, mounted file systems table

SYNOPSIS

/etc/pfs_fstab

/etc/mtab

DESCRIPTION

The **/etc/pfs_fstab** file contains entries for CD-ROM file systems and disc images to mount using the **pfs_mount(1M)** command, which is normally invoked by **rc(1M)** script at boot time. This file is used by various utilities such as **pfs_mount**, and **pfs_umount**.

The **/etc/mtab** file contains entries for file systems *currently* mounted, and is read by programs using the routines described in **getmntent(3)**. **umount** (see **mount(1M)**) removes entries from this file.

Each entry consists of a line of the form:

filesystem directory type options freq pass

filesystem is the pathname of a raw or block-special device, the name of a remote file system in *host:pathname* form, or the name of a file created with MakeDisc.

directory is the pathname of the directory on which to mount the file system.

type is the file system type, which can be one of:

pfs-iso9660 to mount a device as iso9660.

pfs-hsfs to mount a device as hsfs.

pfs-rrip to mount a device as rrip.

pfs-nfs to mount an exported PFS file system

options contains a comma-separated list (no spaces) of mounting options, some of which can be applied to all types of file systems, and others which only apply to specific types.

options valid on *all* file systems:

ro Even if not specified, this option is implied.

suid | nosuid Setuid execution allowed or disallowed.

bg | fg If the first attempt fails, retry in the background, or, in the foreground.

retry=*n* The number of times to retry the mount operation.

rsize=*n* Set the read buffer size to *n* bytes.

timeo=*n* Set the PFS timeout to *n* tenths of a second.

retrans=*n* The number of PFS retransmissions.

soft | hard Return an error if the server does not respond, or continue the retry request until the server responds.

intr Allow keyboard interrupts on hard mounts.

options specific to **iso9660** and **hsfs** file systems:

xlat=xlat_flags

xlat_flags is a colon (:) separated list of translation options. Currently supported are **no_version**, **dot_version**, **lower_case**, and **unix**.

freq is the interval (in days) between dumps. For a PFS file system, this should be 0.

pass is the **fsck(1M)** pass in which to check the partition. For a PFS file system, this should be 0.

A pound-sign (#) as the first character indicates a comment line which is ignored by routines that read this file. The order of records in **/etc/pfs_fstab** is important because **fsck**, **mount**, and **umount** process the file sequentially; an entry for a file system must appear *after* the entry for any file system it is to be mounted on top of.

EXAMPLES

/dev/sr0 /cd-rom pfs-iso9660 ro,suid 0 0

example:/home/user /home/user pfs-nfs ro,hard,fg 0 0

AUTHOR

pfs_fstab was developed by Young Minds, Inc.

FILES

`/etc/pfs_fstab`
`/etc/mtab`

SEE ALSO

`pfs_mount(1M)`.



p

NAME

portal - a "window to the future" for applications

SYNOPSIS

```
#include <sys/portal.h>
```

DESCRIPTION

This header file is a "window to the future" for applications. It will help you to:

- Write code that is portable across 32-bit and 64-bit systems,
- Avoid undocumented assumptions about sizes of integral types,
- Write portable code that needs to be explicit about the sizes of integral types,
- Write code that is portable to a platform which has different sizes of integral types, and
- Share frequently used macros that are portable across 32-bit and 64-bit systems.

In addition to the macros defined in this file, it includes the header files **limits.h** (see *limits(5)*) and **inttypes.h** (see *inttypes(5)*).

The following macros are defined in **sys/portal.h**:

SET_MASK_BIT(*bit_num*, *type*)

This macro can be used to create a mask that has one bit set. *bit_num* is the position of the bit to set, and *type* is the data type of the mask. A -1 is returned in the case of overflow or underflow.

SET_MASK_BIT_LOOP(*mask*, *bit_num*, *type*)

This macro can be used to set a bit in a mask. *mask* is the current value of the mask, *bit_num* is the position of the bit to set, and *type* is the data type of the mask.

SIGN_BIT(*type*)

This macro can be used to return the bit position of the sign bit for the specified data type. *type* is the data type for which to return the position of the sign bit.

SIGN_BIT_MASK(*type*)

This macro can be used to return a mask for the sign bit for the specified data type. *type* is the data type for which to return the sign bit mask.

SIGN_EXTEND(*value*, *old_type*, *new_type*)

This macro can be used to do a sign extension from one data type to another. *value* is the current value that is to be sign-extended. *old_type* is the current data type of *value* and *new_type* is the new data type of *value*.

TEST_ENDIAN(*endian*)

This macro can be used to check if code has been compiled big or little endian. *endian* is an integer in which the result will be returned.

The following macros can be used for print formatting and scan formatting of values of data types that can change in size based upon the compilation flag **_FILE_OFFSET_BITS**. Examples of such data types are *off_t* and *fpos_t*.

```
PRIdF64    d  print formatting option for a 32-bit or 64-bit size value.
PRIOF64    o  print formatting option for a 32-bit or 64-bit size value.
PRIxF64    x  print formatting option for a 32-bit or 64-bit size value.
PRIuF64    u  print formatting option for a 32-bit or 64-bit size value.
```

```
SCNdF64    d  scan formatting option for a 32-bit or 64-bit size value.
SCNoF64    o  scan formatting option for a 32-bit or 64-bit size value.
SCNxF64    x  scan formatting option for a 32-bit or 64-bit size value.
SCNuF64    u  scan formatting option for a 32-bit or 64-bit size value.
```

EXAMPLES

The **SET_MASK_BIT** macro in the following example will turn on the high bit in a 64-bit integer.

```
SET_MASK_BIT(SIGN_BIT(int64_t), int64_t)
```

The **SET_MASK_BIT** macro in the following example will be used to turn on all bits except the sign bit in a 32-bit integer.

```
~SET_MASK_BIT(SIGN_BIT(int32_t), int32_t)
```


The `SET_MASK_BIT_LOOP` macro in the following example will turn on the three least significant bits of the maximum integer.

```
int i;
intmax_t mask = 0;

for (i = 0; i < 3; i++) {
    SET_MASK_BIT_LOOP(mask, i, intmax_t);
}
```

The `SIGN_BIT` macro in the following example will return the position of the sign bit in a 32-bit integer.

```
SIGN_BIT(int32_t)
```

The `SIGN_BIT_MASK` macro in the following example will return a sign bit mask for a 32-bit integer.

```
SIGN_BIT_MASK(int32_t)
```

The `SIGN_EXTEND` macro in the following example will convert the 8-bit integer stored in a char data type to a 64-bit integer and correctly extend the sign.

```
char c;
int64_t i;

i = SIGN_EXTEND(c, char, int64_t);
```

The `TEST_ENDIAN` macro in the following example will store a 1 in *endian* if the compilation was big endian; otherwise, it will store a 0 in *endian*.

```
int endian;

TEST_ENDIAN(endian);

if (endian == 0)
    printf("This a little endian system\n");

if (endian == 1)
    printf("This a big endian system\n");
```

AUTHOR

`portal.h` was developed by HP.

FILES

`/usr/include/sys/portal.h`

SEE ALSO

`inttypes(5)`, `limits(5)`, `printf(3S)`, `scanf(3S)`.

NAME

quota - disk quotas

DESCRIPTION

Disk quotas can be used by the system administrator to limit the number of files and file blocks owned by a user on a per-file-system basis. Separate limits can be established for both the number of files (inodes) and the number of 1-Kbyte blocks for each user. A **soft** (preferred) and a **hard** limit are established.

For example, user **joe_doe** may have soft limits of 1000 blocks and 200 files and hard limits of 1200 blocks and 300 files on the root file system (/) containing his/her **HOME** directory and **/tmp**, and soft and hard block limits of 100 and 120, respectively, with no explicit file limit (0), on the mounted file system **/mnt**.

A time limit is established for each file system which determines how long a user is allowed to exceed the soft limit. The default time limit is one week (7 days).

When a user exceeds his/her soft limit, a warning is emitted on the user's terminal. The user can continue to increase utilization over the soft limit until he/she either exceeds the hard limit or the established time limit. Once either of these events occurs, a message is sent to the user's terminal and further attempts at file creation and/or increased block utilization will fail. At this point, the user must reduce his/her use of the exceeded limit below the soft limit to restore normal operation.

At login time, users exceeding quota limits are reminded (via *login(1)*) of exceeded quotas and appropriate remedial action. The user can check current quota status at any time with the **quota** command (see *quota(1)*).

Quota limits and utilization statistics are maintained by the operating system for each file system for which quotas have been enabled (see *mount(1M)* and *quotaon(1M)*).

Disk quotas are established independently for each user and each file system via the **edquota** command (see *edquota(1M)*). This command is also used to establish the limit for the amount of time users are permitted to exceed their soft limit. The default time limit is 1 week.

Limits and usage statistics are stored statically in the file **quotas** on the root of each file system for which they are in effect. This file is synchronized with information in the kernel by the **quotactl()** system call (see *quotactl(2)*) and whenever an affected file system is unmounted.

Quotas can be enabled automatically at boot or mount time by adding the **quotas** option to the option list in **/etc/fstab** (see *fstab(4)* and *mount(1M)*). By default, **mount** does not enable disk quotas.

Quotas can subsequently be disabled and re-enabled with the **quotaoff** and **quotaon** commands (see *quotaon(1M)*). When quotas are disabled, the kernel does not maintain usage statistics and the **quotas** file usage statistics are invalidated by file system activity. Disabling quotas improves performance, but necessitates running the **quotacheck** command (see *quotacheck(1M)*) to update the kernel and **quotas** file after subsequently re-enabling quotas.

The **repquota** command (see *repquota(1M)*) displays reports of current quota statistics. The somewhat related, but independent, **quot** command (see *quot(1M)*), collects and reports disk utilization independently of the disk quota subsystem.

The **mount** command (see *mount(1M)*) reports any file systems for which quotas are enabled.

Data Storage Structure

The **dqblk** data structure (defined in **<quota.h>**), is used by the **quotactl()** system call (see *quotactl(2)*) to get or set quota information. This structure contains fields that are used to store a user's current file and block count and quota limits for a particular file system.

struct dqblk contains the following members:

```

    u_long    dqb_bhardlimit; /* maximum # of disk blocks +1 */
    u_long    dqb_bsoftlimit; /* preferred limit on disk blocks */
    u_long    dqb_curblocks;  /* current block count */
    u_long    dqb_fhardlimit; /* maximum # allocated files +1 */
    u_long    dqb_fsoftlimit; /* preferred file limit */
    u_long    dqb_curfiles;   /* current # allocated files */
    u_long    dqb_btimelimit; /* time limit for excessive block use */
    u_long    dqb_ftimelimit; /* time limit for excessive files */

```

NETWORKING FEATURES

Quotas are not fully supported over NFS file systems. However, the **quota** command is able to report quota statistics on remote NFS file systems for which disk quotas are in effect, if the remote system provides the RPC **rquotad** service (see *rquotad(1M)*).

rquotad is provided to allow reciprocal support to other systems.

EXAMPLES

Initial Setup

The kernel must be reconfigured to support disk quotas; see the System Administration manuals. Eligible file systems for disk quota enforcement are those with mount options **rw** and **quota**, as described in *mount(1M)* and *fstab(4)*.

For each file system for which quotas are to be enabled, perform the following tasks:

1. Mount the file system.
2. Add **quota** to the existing options list in **/etc/fstab**. For example, change the string **default** for the root (/) entry to **default,quota**. Once this is done, quotas will automatically be enabled for all relevant file systems on system reboot.
3. Create the **quotas** file in the mount directory of the file system. For example, for the **/mnt** file system, run the command


```
cpset /dev/null /mnt/quotas 600 root bin
```
4. Establish one or more prototype user quotas using the **edquota** command (see *edquota(1M)*).
If you want a number of users on your system to have the same limits, use **edquota** to set those quotas for a prototype user; then use the **edquota -p** command to replicate those limits for that group of users.
5. Turn on the quotas on the file system using **quotaon**. For example, run the command


```
/usr/sbin/quotaon /mnt
```
6. Run **quotacheck** (see *quotacheck(1M)*) on the file system to record the current usage statistics.

Adding a new user

To add a new user to the quota system:

1. Use **edquota** to copy the quotas of an existing user.
2. Run **quotacheck**.

Adding a new file system to an established system

Repeat steps 1 through 5 above under "Initial Setup" for the new file system.

WARNINGS

The HP-UX default is to allow **chown(2)**. This can interfere with the disk quota mechanism. Quotas can be defeated if the **chown** command (see *chown(1)*) or the **chown()** system call (see *chown(2)*) is accessible to a user. The **setprivgrp** command (see *setprivgrp(1M)*) can be used to limit access to the **chown()** system call so that only a specified group of users are permitted to use the **chown** command or the **chown()** system call.

The **sam** command (see *sam(1M)*) does not yet support disk quotas. When adding new users or file systems, any desired quotas must be established outside of **sam**.

HP has added features to the original implementation to ensure correctness of the content of the quotas file when quotas are enabled by **mount** and disabled by **umount** (see *mount(1M)*), thus eliminating the need to run **quotacheck** (see *quotacheck(1M)*). These features are ineffective, however, if **quotaoff** and **quotaon** (see *quotaon(1M)*) are used to control quotas.

quotacheck should only be run on a dormant file system to ensure accurate usage information. The **-qv** options of the **fsclean** command (see *fsclean(1M)*) report on the the current viability of the quota information.

Currently, only HFS file systems support disk quotas, though there may be support for other file systems in the future.

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, and HP.

FILES

<code>/etc/fstab</code>	Static information about the file systems
<code>/etc/mnttab</code>	Mounted file system table
<code>directory/quotas</code>	Quota statistics static storage for a file system, where <i>directory</i> is the root of the file system, as specified to the <code>mount</code> command (see <code>mount(1M)</code>).

SEE ALSO

`chown(1)`, `chown(2)`, `quota(1)`, `edquota(1M)`, `fstab(4)`, `mount(1M)`, `quot(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `rquotad(1M)`, `setprivgrp(1M)`, `quotactl(2)`, `vfsmount(2)`.

NAME

rcsintro - description of RCS commands

DESCRIPTION

Revision Control System (RCS) automates the storing, retrieval, logging, identification, and merging of revisions of ASCII text files. RCS is useful for managing files that are revised frequently.

Functions of RCS

- Storage and retrieval of revisions of text files. RCS saves revisions in a space efficient way. Revisions can be retrieved by ranges of revision numbers, symbolic names, dates, authors, and states.
- Maintenance of a complete history of changes. RCS logs all changes automatically. In addition to the text of each revision, RCS stores the author, date and time of check in, and a log message summarizing the change.
- Resolution of access conflicts. When two or more people try to modify the same revision of a file, RCS alerts them and prevents one modification from corrupting the other.
- Maintenance of a tree of revisions. RCS can maintain separate lines of development for each file. It stores a tree structure that represents the ancestral relationships among revisions.
- Merging of revisions and resolution of conflicts. Two separate lines of development of a file can be coalesced by merging. If the revisions to be merged affect the same lines of a file, RCS flags the overlapping changes.
- Release and configuration control. Revisions can be assigned symbolic names and marked as released, stable, experimental, etc. With these facilities, configurations of a file can be described simply and directly.
- Automatic identification of each revision with filename, revision number, creation time, author, etc. This identification is like a stamp that can be embedded at an appropriate place in the text of a revision. These stamps make it simple to determine which revisions of which files make up a given configuration.
- Minimization of secondary storage. RCS uses very little extra space for revisions (only the differences are stored). If intermediate revisions are deleted, the remaining deltas are compressed accordingly.

Getting Started with RCS

The basic user interface is extremely simple. The novice only needs to learn two commands: *ci*(1) and *co*(1). *ci*, short for "check in," deposits the contents of a text file into an archival file called an RCS file. An RCS file contains all revisions of a particular text file. *co*, short for "check out", retrieves revisions from an RCS file.

Suppose you have a file **f.c** that you wish to put under control of RCS. Invoke the check in command:

ci f.c

This command creates the RCS file **f.c,v**, stores **f.c** into it as revision **1.1**, and deletes **f.c**. It also asks you for a description. The description should be a synopsis of the contents of the file. All subsequent check-in commands will ask for a log entry, which should summarize the changes that were made.

Files with names ending with **,v** are called RCS files ("**v**" stands for "versions"), all other files are presumed to be working files. To get back the working file **f.c** in the previous example, use the check out command:

co f.c

This command extracts the latest revision from **f.c,v** and writes it into **f.c**. You can now edit **f.c** and check it back in by invoking:

ci f.c

ci increments the revision number properly. If *ci* complains with the message:

ci error: no lock set by <your login>

your system administrator has decided to create all RCS files with the locking attribute set to "strict". In this case, you should have locked the revision during the previous check out. Your last check out should have been:

co -l f.c

Of course, it is too late now to do the check out with locking, because you probably modified **f.c** already, and a second check out would overwrite your modifications. Instead, invoke:

r_{cs} -l f.c

This command will lock the latest revision for you, unless somebody else has already locked it. In that case, you will have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, etc. All that locking prevents is a check in by anybody but the locker.

If your RCS file is private, i.e., if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for check in; all others still do. Turning strict locking off and on is done with the commands:

r_{cs} -U f.c

and

r_{cs} -L f.c

If you do not want to clutter your working directory with RCS files, create a subdirectory called RCS in your working directory, and move all your RCS files there. RCS commands will search that directory to find needed files. All the commands discussed above will still work without any modification.

To avoid the deletion of the working file during check in (in case you want to continue editing), invoke:

c_i -l f.c

or

c_i -u f.c

These commands check in **f.c** as usual, but perform an implicit check out. The first form also locks the checked in revision, the second one does not. Thus, these options save you one check out operation. The first form is useful if locking is strict; the second one if not strict. Both update the identification markers in your working file (see below).

You can give *ci* the number you want assigned to a checked in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. The command:

c_i -r2 f.c

or

c_i -r2.1 f.c

assigns the number 2.1 to the new revision. From then on, *ci* will number the subsequent revisions with 2.2, 2.3, etc. The corresponding *co* commands:

c_o -r2 f.c

and

c_o -r2.1 f.c

retrieve the latest revision numbered 2.x and the revision 2.1, respectively. *co* without a revision number selects the latest revision on the "trunk"; i.e., the highest revision with a number consisting of 2 fields. Numbers with more than 2 fields are needed for branches. For example, to start a branch at revision 1.3, invoke:

c_i -r1.3.1 f.c

This command starts a branch numbered 1 at revision 1.3, and assigns the number 1.3.1.1 to the new revision. For more information about branches, see *r_{cs}file(4)*.

RCS File Naming and Location

RCS recognizes two kinds of files: RCS files (revision archives), and working files. Working filenames are defined by the RCS user, RCS file names are generated by RCS by appending ",v" to the working file name. Pairs of RCS files and working files can be specified in 3 ways:

- Both the RCS file and the working file are given. The RCS filename is of the form **path1/workfile,v** and the working filename is of the form **path2/workfile**, where **path1** and **path2** are (possibly different or empty) paths and **workfile** is a filename.
- Only the RCS file is given. Then the working file is assumed to be in the current directory and its name is derived from the name of the RCS file by removing **path1/** and the suffix **",v"**.

- Only the working file is given. Then the name of the RCS file is derived from the name of the working file by removing **path2/** and appending the suffix **,"v"**.

If the RCS filename is omitted or specified without a path, RCS commands look for the RCS file in the directory **./RCS** (or the directory it points to if it is a directory link), then in the current working directory.

RCS Directory Links

RCS supports directory links. If a regular file named RCS exists in the current working directory, RCS interprets the first line as a path name to the directory where RCS files are stored. RCS can follow a chain of up to ten directory links to reach the RCS directory.

Automatic Identification

RCS can put special strings for identification into your source and object code. To obtain such identification, place the marker:

\$Header\$

into your text, for instance inside a comment. RCS replaces this marker with a string of the form:

\$Header: *filename revision_number date time author state\$*

With such a marker on the first page of each module, you can always see with which revision you are working. RCS keeps the markers up-to-date automatically. To propagate the markers into your object code, simply put them into literal character strings. In C, this is done as follows:

```
static char rcsid[] = $Header$ ;
```

The command *ident* extracts such markers from any file, even object code and dumps. Thus, *ident* lets you find out which revisions of which modules were used in a given program.

You may also find it useful to put the marker **\$Log\$** into your text, inside a comment. This marker accumulates the log messages that are requested during check in. Thus, you can maintain the complete history of your file directly inside it. There are several additional identification markers. See *co(1)* for details.

WARNINGS

Names of RCS files are generated by appending **,"v"** to the end of the working file name. If the resulting RCS file name is too long for the file system on which the RCS file should reside, the RCS command terminates with an error message.

RCS is designed to be used with TEXT files only. Attempting to use RCS with non-text (binary) files will result in data corruption.

AUTHOR

rcsintro was developed by Walter F. Tichy, Purdue University, West Lafayette, IN 47907.
Revision Number: 3.0; Release Date: 83/05/11.
Copyright 1982 by Walter F. Tichy.

SEE ALSO

ci(1), *co(1)*, *ident(1)*, *merge(1)*, *rcs(1)*, *rcsdiff(1)*, *rcsmerge(1)*, *rlog(1)*, *rcsfile(4)*.

Walter F. Tichy, "Design, Implementation, and Evaluation of a Revision Control System," in *Proceedings of the 6th International Conference on Software Engineering*, IEEE, Tokyo, Sept. 1982.

r

NAME

regex - regular expression and pattern matching notation definitions

DESCRIPTION

A **regular expression** is a mechanism supported by many utilities for locating and manipulating patterns in text. **pattern matching notation** is used by shells and other utilities for file name expansion. This manual entry defines two forms of regular expressions: **Basic Regular Expressions** and **Extended Regular Expressions**; and the one form of **Pattern Matching Notation**.

BASIC REGULAR EXPRESSIONS

Basic regular expression (RE) notation and construction rules apply to utilities defined as using basic REs. Any exceptions to the following rules are noted in the descriptions of the specific utilities that use REs.

REs Matching a Single Character

The following REs match a single character or a single collating element:

Ordinary Characters

An ordinary character is an RE that matches itself. An ordinary character is any character in the supported character set except <newline> and the regular expression special characters listed in Special Characters below. An ordinary character preceded by a backslash (\) is treated as the ordinary character itself, except when the character is (,), {, or }, or the digits 1 through 9 (see REs Matching Multiple Characters). Matching is based on the bit pattern used for encoding the character; not on the graphic representation of the character.

Special Characters

A regular expression special character preceded by a backslash is a regular expression that matches the special character itself. When not preceded by a backslash, such characters have special meaning in the specification of REs. Regular expression special characters and the contexts in which they have special meaning are:

. [\	The period, left square bracket, and backslash are special except when used in a bracket expression (see RE Bracket Expression).
*	The asterisk is special except when used in a bracket expression, as the first character of a regular expression, or as the first character following the character pair \ ((see REs Matching Multiple Characters).
^	The circumflex is special when used as the first character of an entire RE (see Expression Anchoring) or as the first character of a bracket expression.
\$	The dollar sign is special when used as the last character of an entire RE (see Expression Anchoring).
<i>delimiter</i>	Any character used to bound (i.e., delimit) an entire RE is special for that RE.

Period

A period (.), when used outside of a bracket expression, is an RE that matches any printable or nonprintable character except <newline>.

RE Bracket Expression

A bracket expression enclosed in square brackets ([]) is an RE that matches a single collating element contained in the nonempty set of collating elements represented by the bracket expression.

The following rules apply to bracket expressions:

bracket expression

A bracket expression is either a **matching list expression** or a **non-matching list expression**, and consists of one or more expressions in any order. Expressions can be: collating elements, collating symbols, noncollating characters, equivalence classes, range expressions, or character classes. The right bracket (]) loses its special meaning and represents itself in a bracket expression if it occurs first in the list (after an initial ^, if any). Otherwise, it terminates the bracket expression (unless it is the ending right bracket for a valid collating symbol, equivalence class, or character class, or it is the collating element within a collating symbol or equivalence class expression). The special characters

. * [\

(period, asterisk, left bracket, and backslash) lose their special meaning within a bracket expression.

The character sequences:

[. [= [:

(left-bracket followed by a period, equal-sign or colon) are special inside a bracket expression and are used to delimit collating symbols, equivalence class expressions and character class expressions. These symbols must be followed by a valid expression and the matching terminating .], =], or :].

matching list A matching list expression specifies a list that matches any one of the characters represented in the list. The first character in the list cannot be the circumflex. For example, **[abc]** is an RE that matches any of **a**, **b**, or **c**.

non-matching list

A **non-matching list** expression begins with a circumflex (^), and specifies a list that matches any character or collating element *except* <newline> and the characters represented in the list. For example, **[^abc]** is an RE that matches any character except <newline> or **a**, **b**, or **c**. The circumflex has this special meaning *only* when it occurs first in the list, immediately following the left square bracket.

collating element

A **collating element** is a sequence of one or more characters that represents a single element in the collating sequence as identified via the most current setting of the locale category LC_COLLATE (see *setlocale*(3C)).

collating symbol

A **collatingsymbol** is a collating element enclosed within bracket-period ([...]) delimiters. Multi-character collating elements must be represented as collating symbols to distinguish them from single-character collating elements. For example, if the string **ch** is a valid collating element, then **[[.ch.]]** is treated as an element matching the same string of characters, while **ch** is treated as a simple list of the characters **c** and **h**. If the string within the bracket-period delimiters is not a valid collating element in the current collating sequence definition, the symbol is treated as an invalid expression.

noncollating character

A **noncollating character** is a character that is ignored for collating purposes. By definition, such characters cannot participate in equivalence classes or range expressions.

equivalence class

An **equivalence class** expression represents the set of collating elements belonging to an equivalence class. It is expressed by enclosing any one of the collating elements in the equivalence class within bracket-equal ([=...=]) delimiters. For example, if **a**, **á**, and **A** belong to the same equivalence class, then **[[=a=]b]**, **[[=á=]b]**, and **[[=A=]b]** are each equivalent to **[aáAb]**.

range expression

A **range expression** represents the set of collating elements that fall between two elements in the current collation sequence as defined via the most current setting of the locale category LC_COLLATE (see *setlocale*(3C)). It is expressed as the starting point and the ending point separated by a hyphen (-).

The starting range point and the ending range point must be a collating element, collating symbol, or equivalence class expression. An *equivalence class expression* used as an end point of a range expression is interpreted such that all collating elements within the equivalence class are included in the range. For example, if the collating order is **A**, **a**, **B**, **b**, **C**, **c**, **ch**, **D**, and **d** and the characters **A** and **a** belong to the same equivalence class, then the expression **[[=a=-]D]** is treated as **[AaBbCc.ch.D]**.

Both starting and ending range points must be valid collating elements, collating symbols, or equivalence class expressions, and the ending range point must collate equal to or higher than the starting range point; otherwise the expression is invalid. For example, with the above collating order and assuming that **E** is a noncollating

I

character, then both the expressions `[[=A=]-E]` and `[d-a]` are invalid.

An ending range point can also be the starting range point in a subsequent range expression. Each such range expression is evaluated separately. For example, the bracket expression `[a-m-o]` is treated as `[a-mm-o]`.

The hyphen character is treated as itself if it occurs first (after an initial `^`, if any) or last in the list, or as the rightmost symbol in a range expression. As examples, the expressions `[-ac]` and `[ac-]` are equivalent and match any of the characters `a`, `c`, or `-`; the expressions `[^-ac]` and `[^ac-]` are equivalent and match any characters except `<newline>`, `a`, `c`, or `-`; the expression `[%--]` matches any of the characters in the defined collating sequence between `%` and `-` inclusive; the expression `[-@]` matches any of the characters in the defined collating sequence between `-` and `@` inclusive; and the expression `[a--@]` is invalid, assuming `-` precedes `a` in the collating sequence.

If a bracket expression must specify both `-` and `]`, the `]` must be placed first (after the `^`, if any) and the `-` last within the bracket expression.

character class

A character class expression represents the set of characters belonging to a character class, as defined via the most current setting of the locale category `LC_CTYPE`. It is expressed as a character class name enclosed within bracket-colon (`[:]`) delimiters.

Standard character class expressions supported in all locales are:

[alpha:]	letters
[upper:]	upper-case letters
[lower:]	lower-case letters
[digit:]	decimal digits
[xdigit:]	hexadecimal digits
[alnum:]	letters or decimal digits
[space:]	characters producing white-space in displayed text
[print:]	printing characters
[punct:]	punctuation characters
[graph:]	characters with a visible representation
[cntrl:]	control characters
[blank:]	blank characters

REs Matching Multiple Characters

The following rules may be used to construct REs matching multiple characters from REs matching a single character:

<i>RE</i> <i>RE</i>	The concatenation of REs is an RE that matches the first encountered concatenation of the strings matched by each component of the RE. For example, the RE bc matches the second and third characters of the string abcdefabcdef .
<i>RE</i> *	An RE matching a single character followed by an asterisk (*) is an RE that matches zero or more occurrences of the RE preceding the asterisk. The first encountered string that permits a match is chosen, and the matched string will encompass the maximum number of characters permitted by the RE. For example, in the string abbbcd eabbbbbbcd e , both the RE b*c and the RE bbb*c are matched by the substring bbbc in the second through fifth positions. An asterisk as the first character of an RE loses this special meaning and is treated as itself.
<code>\(RE\)</code>	A subexpression can be defined within an RE by enclosing it between the character pairs <code>\(</code> and <code>\)</code> . Such a subexpression matches whatever it would have matched without the <code>\(</code> and <code>\)</code> . Subexpressions can be arbitrarily nested. An asterisk immediately following the <code>\(</code> loses its special meaning and is treated as itself. An asterisk immediately following the <code>\)</code> is treated as an invalid character.
<code>\n</code>	The expression <code>\n</code> matches the same string of characters as was matched by a subexpression enclosed between <code>\(</code> and <code>\)</code> preceding the <code>\n</code> . The character <i>n</i> must be a

digit from **1** through **9**, specifying the *n*-th subexpression (the one that begins with the *n*-th `\(` and ends with the corresponding paired `\)`). For example, the expression `^\(.*\)1$` matches a line consisting of two adjacent appearances of the same string.

If the `\n` is followed by an asterisk, it matches zero or more occurrences of the subexpression referred to. For example, the expression `\(ab\(\cd\)ef\)Z\2*Z\1` matches the string **abcdefZcdcdZabcdef**.

RE`\{m,n\}` An RE matching a single character followed by `\{m\}`, `\{m,\}`, or `\{m,n\}` is an RE that matches repeated occurrences of the RE. The values of *m* and *n* must be decimal integers in the range 0 through 255, with *m* specifying the exact or minimum number of occurrences and *n* specifying the maximum number of occurrences. `\{m\}` matches exactly *m* occurrences of the preceding RE, `\{m,\}` matches at least *m* occurrences, and `\{m,n\}` matches any number of occurrences between *m* and *n*, inclusive.

The first encountered string that matches the expression is chosen; it will contain as many occurrences of the RE as possible. For example, in the string **abbbbbbbbc** the RE **b\{3\}** is matched by characters two through four, the RE **b\{3,\}** is matched by characters two through eight, and the RE **b\{3,5\}**c is matched by characters four through nine.

Expression Anchoring

An RE can be limited to matching strings that begin or end a line (i.e., anchored) according to the following rules:

- A circumflex (`^`) as the first character of an RE anchors the expression to the beginning of a line; only strings starting at the first character of a line are matched by the RE. For example, the RE `^ab` matches the string **ab** in the line **abcdef**, but not the same string in the line **cdefab**.
- A dollar sign (`$`) as the last character of an RE anchors the expression to the end of a line; only strings ending at the last character of a line are matched by the RE. For example, the RE `ab$` matches the string **ab** in the line **cdefab**, but not the same string in the line **abcdef**.
- An RE anchored by both `^` and `$` matches only strings that are lines. For example, the RE `^abcdef$` matches only lines consisting of the string **abcdef**.

The use of duplication characters (`+`, `*`) following anchors is illegal.

EXTENDED REGULAR EXPRESSIONS

The extended regular expression (ERE) notation and construction rules apply to utilities defined as using extended REs. Any exceptions to the following rules are noted in the descriptions of the specific utilities using EREs.

EREs Matching a Single Character

The following EREs match a single character or a single collating element:

Ordinary Characters

An ordinary character is an ERE that matches itself. An ordinary character is any character in the supported character set except `<newline>` and the regular expression special characters listed in Special Characters below. An ordinary character preceded by a backslash (`\`) is treated as the ordinary character itself. Matching is based on the bit pattern used for encoding the character, not on the graphic representation of the character.

Special Characters

A regular expression special character preceded by a backslash is a regular expression that matches the special character itself. When not preceded by a backslash, such characters have special meaning in the specification of EREs. The extended regular expression special characters and the contexts in which they have their special meaning are:

- `[\ () * + ? $ |` The period, left square bracket, backslash, left parenthesis, right parenthesis, asterisk, plus sign, question mark, dollar sign, and vertical bar are special except when used in a bracket expression (see ERE Bracket Expression).
- `^` The circumflex is special except when used in a bracket expression in a non-leading position.
- *delimiter* Any character used to bound (i.e., delimit) an entire ERE is special for that ERE.

Period

A period (`.`), when used outside of a bracket expression, is an ERE that matches any printable or nonprintable character except `<newline>`.

ERE Bracket Expression

The syntax and rules for ERE bracket expressions are the same as for RE bracket expressions found above.

EREs Matching Multiple Characters

The following rules may be used to construct EREs matching multiple characters from EREs matching a single character:

<i>RE</i> <i>RE</i>	A concatenation of EREs matches the first encountered concatenation of the strings matched by each component of the ERE. Such a concatenation of EREs enclosed in parentheses matches whatever the concatenation without the parentheses matches. For example, both the ERE bc and the ERE (bc) matches the second and third characters of the string abcdefabcdef . The longest overall string is matched.
<i>RE</i> +	The special character plus (+), when following an ERE matching a single character, or a concatenation of EREs enclosed in parenthesis, is an ERE that matches one or more occurrences of the ERE preceding the plus sign. The string matched will contain as many occurrences as possible. For example, the ERE b+c matches the fourth through seventh characters in the string acabbbcd .
<i>RE</i> *	The special character asterisk (*), when following an ERE matching a single character, or a concatenation of EREs enclosed in parenthesis, is an ERE that matches zero or more occurrences of the ERE preceding the asterisk. For example, the ERE b*c matches the first character in the string cabbbcd . If there is any choice, the longest left-most string that permits a match is chosen. For example, the ERE b*cd matches the third through seventh characters in the string cabbbcd .
<i>RE</i> ?	The special character question mark (?), when following an ERE matching a single character, or a concatenation of EREs enclosed in parenthesis, is an ERE that matches zero or one occurrences of the ERE preceding the question mark. The string matched will contain as many occurrences as possible. For example, the ERE b?c matches the second character in the string acabbbcd .
<i>RE</i> { <i>m,n</i> }	interval expression that functions the same way as basic regular expression syntax, <i>RE</i> { <i>m,n</i> }

Alternation

Two EREs separated by the special character vertical bar (**|**) matches a string that is matched by either ERE. For example, the ERE **((ab)|c)d** matches the string **abd** and the string **cd**.

Precedence

The order of precedence is as follows, from high to low:

[]	square brackets
* + ?	asterisk, plus sign, question mark
^ \$	anchoring
	concatenation
	alternation

For example, the ERE **abba|cde** is interpreted as "match either **abba** or **cde**. It does not mean "match **abb** followed by **a** or **c** followed in turn by **de** (because concatenation has a higher order of precedence than alternation).

Expression Anchoring

An ERE can be limited to matching strings that begin or end a line (i.e., anchored) according to the following rules:

- A circumflex (**^**) matches the beginning of a line (anchors the expression to the beginning of a line). For example, the ERE **^ab** matches the string **ab** in the line **abcdef**, but not the same string in the line **cdefab**.

- A dollar sign (\$) matches the end of a line (anchors the expression to the end of a line). For example, the ERE **ab\$** matches the string **ab** in the line **cdefab**, but not the same string in the line **abcdef**.
- An ERE anchored by both ^ and \$ matches only strings that are lines. For example, the ERE **^abcdef\$** matches only lines consisting of the string **abcdef**. Only empty lines match the ERE **^\$**.

The use of duplication characters (+,*) following anchors is illegal.

PATTERN MATCHING NOTATION

The following rules apply to pattern matching notation except as noted in the descriptions of the specific utilities using pattern matching.

Patterns Matching a Single Character

The following patterns match a single character or a single collating element:

Ordinary Characters

An ordinary character is a pattern that matches itself. An ordinary character is any character in the supported character set except <newline> and the pattern matching special characters listed in Special Characters below. Matching is based on the bit pattern used for encoding the character, not on the graphic representation of the character.

Special Characters

A pattern matching special character preceded by a backslash (\) is a pattern that matches the special character itself. When not preceded by a backslash, such characters have special meaning in the specification of patterns. The pattern matching special characters and the contexts in which they have their special meaning are:

? * [The question mark, asterisk, and left square bracket are special except when used in a bracket expression (see Pattern Bracket Expression).

Question Mark

A question mark (?), when used outside of a bracket expression, is a pattern that matches any printable or nonprintable character except <newline>.

Pattern Bracket Expression

The syntax and rules for pattern bracket expressions are the same as for RE bracket expressions found above with the following exceptions:

The exclamation point character (!) replaces the circumflex character (^) in its role in a non-matching list in the regular expression notation.

The backslash is used as an escape character within bracket expressions.

Patterns Matching Multiple Characters

The following rules may be used to construct patterns matching multiple characters from patterns matching a single character:

***** The asterisk (*) is a pattern that matches any string, including the null string.

RE RE The concatenation of patterns matching a single character is a valid pattern that matches the concatenation of the single characters or collating elements matched by each of the concatenated patterns. For example, the pattern **a[bc]** matches the string **ab** and **ac**.

 The concatenation of one or more patterns matching a single character with one or more asterisks is a valid pattern. In such patterns, each asterisk matches a string of zero or more characters, up to the first character that matches the character following the asterisk in the pattern.

 For example, the pattern **a*d** matches the strings **ad**, **abd**, and **abcd**; but not the string **abc**. When an asterisk is the first or last character in a pattern, it matches zero or more characters that precede or follow the characters matched by the remainder of the pattern. For example, the pattern **a*d*** matches the strings **ad**, **abcd**, **abcdef**, **aaaad**, and **adddd**; the pattern ***a*d** matches the strings **ad**, **abcd**, **efabcd**, **aaaad**, and **adddd**.

Rule Qualification for Patterns Used for Filename Expansion

The rules described above for pattern matching are qualified by the following rules when the pattern matching notation is used for filename expansion by *sh*(1), *csh*(1), *ksh*(1), and *make*(1).

If a filename (including the component of a pathname that follows the slash (/) character) begins with a period (.), the period must be explicitly matched by using a period as the first character of the pattern; it cannot be matched by either the asterisk special character, the question mark special character, or a bracket expression. This rule does not apply to *make*(1).

The slash character in a pathname must be explicitly matched by using a slash in the pattern; it cannot be matched by either the asterisk special character, the question mark special character, or a bracket expression. For *make*(1) only the part of the pathname following the last slash character can be matched by a special character. That is, all special characters preceding the last slash character lose their special meaning.

Specified patterns are matched against existing filenames and pathnames, as appropriate. If the pattern matches any existing filenames or pathnames, the pattern is replaced with those filenames and pathnames, sorted according to the collating sequence in effect. If the pattern does not match any existing filenames or pathnames, the pattern string is left unchanged.

If the pattern begins with a tilde (~) character, all of the ordinary characters preceding the first slash (or all characters if there is no slash) are treated as a possible login name. If the login name is null (i.e., the pattern contains only the tilde or the tilde is immediately followed by a slash), the tilde is replaced by a pathname of the process's home directory, followed by a slash. Otherwise, the combination of tilde and login name are replaced by a pathname of the home directory associated with the login name, followed by a slash. If the system cannot identify the login name, the result is implementation-defined. This rule does not apply to *sh*(1) or *make*(1).

If the pattern contains a \$ character, variable substitution can take place. Environmental variables can be embedded within patterns as:

`$name`

or:

`${name}`

Braces are used to guarantee that characters following *name* are not interpreted as belonging to *name*. Substitution occurs in the order specified only once; that is, the resulting string is not examined again for new names that occurred because of the substitution.

Rule Qualification for Patterns Used in the case Command

The rules described above for pattern matching are qualified by the following rule when the pattern matching notation is used in the case command of *sh*(1) and *ksh*(1).

Multiple alternative patterns in a single clause can be specified by separating individual patterns with the vertical bar character (|); strings matching any of the patterns separated this way will cause the corresponding command list to be selected.

SEE ALSO

ksh(1), *sh*(1), *fnmatch*(3C), *glob*(3C), *regcomp*(3C), *setlocale*(3C), *environ*(5).

STANDARDS CONFORMANCE

<regex.h>: AES, SVID2, SVID3, XPG2, XPG3, XPG4

(Hewlett-Packard Company)

NAME

sd - create, distribute, install, monitor, and manage software

SYNOPSIS

```
sw<task> [XToolkit Options] [-r] [-d] [-i] [-l] [-p] [-R] [-u] [-v] [-V]
[-a attribute] [-c catalog] [-C session_file] [-D acl_entry] [-f software_file] [-F acl_file]
[-J jobid] [-l level] [-M acl_entry] [-Q date] [-s source] [-S session_file]
[-t target_file] [-x option=value] [-X option_file] [software_selections] [@ target_selections]
```

Remarks

- HP-UX's software distributor, SD-UX, is included with the HP-UX Operating System and manages software on the *local* host only.
- To install and manage software simultaneously on multiple *remote* hosts (including HP-UX, other UNIX® platforms, Windows NT®, and PCs) from a central controller, you must purchase the *HP OpenView Software Distributor* which provides extended software management capabilities. Information specific *only* to the OpenView product is marked with a heading similar to the following:

The following information applies to HP OpenView Software Distributor only.

DESCRIPTION

The SD commands are:

- **sd** - create and monitor software jobs (*HP OpenView Software Distributor only*).
- **swacl** - modify the Access Control Lists (ACLs) which protect software products.
- **swagentd** - serve local or remote SD software management tasks.
- **swask** - ask for a user response.
- **swcluster** - configure diskless clients (HP-UX 10.* only).
- **swconfig** - configure, unconfigure, or reconfigure installed software.
- **swcopy** - copy software products for subsequent installation or distribution.
- **swgettools** - retrieve the SD product from new media.
- **swinstall** - install and configure software products.
- **swjob** - monitor job progress and log files. (*HP OpenView Software Distributor only*).
- **swlist** - display information about software products.
- **swmodify** - modify software product information in a target root or depot.
- **swpackage** - package software products into a distribution directory or tape.
- **swreg** - register or unregister depots or roots.
- **swremove** - unconfigure and remove software products.
- **swverify** - verify software products.

Related commands include:

- **mk_kernel** - build a bootable HP-UX kernel (HP-UX only).
- **pushAgent** - install the HP OpenView Software Distributor agent on remote systems.

The following sections highlight the features that these commands support.

Interactive Operation

By default, all SD commands except **swask** operate in a non-interactive mode. However, **swcopy**, **swinstall**, **swlist**, and **swremove** commands also support a graphical user interface (GUI). To invoke the GUIs, enter the command without any command-line options or use the **-i** option.

The command-line versions of **swinstall** and **swconfig** can be interactive if the **ask** option is set to **true**.

(Hewlett-Packard Company)

The following information applies to HP OpenView Software Distributor only.

To invoke the HP OpenView Software Distributor GUI, use **sd** command. This provides the central interactive interface for creating and monitoring software jobs.

Distributed Operation

All of the commands except **swask**, **swpackage**, and **swmodify** use a distributed model of operation. The commands act as the controller for distributed operations, managing the specific software management tasks. For each *target_selection*, an SD agent process performs the task:

- **swagent** - perform software management tasks as the agent of an SD command.

Communication between the command and each agent, plus other target host activities are facilitated by an SD daemon process:

- **swagentd** - serve local or remote software management tasks.

The following PC information applies only to HP OpenView Software Distributor.

For each PC controller *target_selection*, a single Windows application combines the **swagent** and **swagentd** functionality:

- **SWAGENTD.EXE** - perform software management tasks, serve local PC software for distribution.

Each PC running the **SWAGENTD.EXE** is a PC controller. When distributing PC software, it acts as a fanout server to PC targets. These targets run SD PC agent programs to perform the actual software installation tasks.

Software Job Management

The following paragraph applies only to HP OpenView Software Distributor.

Most SD commands create job information that records the job definition (in a session file), status and log information for the job. Jobs can be executed immediately, or scheduled for later execution. The user can browse the scheduled, active, and completed jobs using either the command line or interactive interfaces.

Secure Operation

SD uses Access Control Lists (ACLs) to authorize a user attempting to create, modify, or read software products in a depot or installed to a root file system. The superuser can grant specific local and remote users specific access permissions to a target host, a target depot, and/or a target root file system.

Because files are loaded and scripts are run as superuser, granting write permission (to install software) on a root file system or insert permission (to create a new root) on a host, effectively gives the user superuser privileges.

SD uses a method based on credentials and passwords to authenticate the user and the SD command performing a given operation.

Flexible Policy Control

Many policies and behaviors for the SD commands can be controlled via the appropriate command options. Options can be defined in an SD *defaults* file, specified on the command-line invocation of a command, or specified in the GUI.

Preview, Diagnostics and Logging

All commands except **swlist** and **swjob** log major events on the controller host and detailed events on the target hosts.

If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. Refer to the **swagent(1M)** **source_depot_audit** option for more information.

The following paragraph applies only to HP OpenView Software Distributor.

You can use the SD interactive interface (invoked using the **sd** command) and the **swjob** command line interface to monitor job progress and to view controller and target log files.

The **swconfig**, **swcopy**, **swinstall**, **swmodify**, **swpackage**, and **swremove** commands support a preview mode, where the commands will proceed through the analysis phase, then exit.

(Hewlett-Packard Company)

The preview mode only applies to non-interactive operations, since the GUIs wait for confirmation after analysis. In the interactive mode, you can resolve invalid conditions that the commands discover before they actually begin loading or removing files.

Software Products

Software products are organized in a multi-level hierarchy: *bundles*, *products*, *subproducts*, and *filesets*. The actual files that make up a product are packaged into filesets. The *software_selections* for an SD command can specify bundles, products, individual subproducts, and/or individual filesets.

Compatible Software

Software products specify what machine types and operating systems they support (i.e. are compatible with). The **swconfig**, **swinstall**, and **swverify** commands can detect and/or enforce the use of compatible software.

Dependencies Between Software

The **swask**, **swconfig**, **swcopy**, **swinstall**, **swremove**, and **swverify** commands support dependencies between filesets and other filesets and products.

If a *software_selection* specifies a dependency on other filesets and/or products, the commands will automatically select that software. An exception is **swremove**, which can automatically select dependent software (filesets and/or products that depend on the *software_selections*).

By default, all dependencies must be resolved before a command will proceed. The user can override this policy using the **enforce_dependencies** option.

SD supports two types of dependencies: *prerequisites* that must be installed and configured before the dependent fileset is installed and configured (respectively); and *corequisites* that must be installed and configured before the dependent is usable..

Product Location and Multiple Versions

The **swinstall** command can install a software product to an alternate product location instead of the default product directory specified by the vendor. (This directory location is the root directory of all the product's files.)

The **swinstall** command can also install multiple versions of a software product to a single target system, each in a unique product location.

The software management commands, **swconfig**, **swlist**, **swremove**, and **swverify** allow a user to select a specific product from the multiple installed versions by specifying the product location as part of the *software_selection*.

Alternate Root Directory and Depot Directory

By default, the **swinstall**, **swlist**, **swverify**, and **swremove** commands operate on the primary root file system of a target host, namely */*. The user can specify an alternate root directory to these commands, meaning a directory other than */* that will eventually be the root of some target host (e.g. building a test system by mounting its root file system).

The **swconfig** command only operates on software installed to the primary root file system, */*.

When operating on a depot, the **swpackage**, **swcopy**, **swlist**, **swverify**, and **swremove** commands by default use the depot located at */var/spool/sw*. The user can also specify an alternate depot directory to these commands.

Disk Space Analysis

The **swcopy**, **swinstall**, and **swpackage** commands perform a disk space analysis on the *target_selections* to ensure that enough free disk space is available to perform the task.

Before performing any disk space analysis, these commands (and also **swverify** and **swremove**) execute the **mount(1M)** command to mount all file systems listed in each target's file system table (*/etc/fstab* or equivalent). This ensures that files are not loaded into a directory below a future mount point. The user can override this mounting policy using the **mount_all_filesystems** option.

Control Scripts

The **swask**, **swconfig**, **swinstall**, **swremove**, and **swverify** commands execute vendor-defined control scripts to perform checks and/or other tasks beyond those explicitly performed by the commands.

The **swask** command can run request scripts to request user responses. The **swinstall** and **swconfig** commands can also run request scripts.

For **swinstall** and **swremove**, a fileset and/or product can include a check script to perform an analysis of each *target_selection* (target host). If this analysis fails, the script can: prevent the fileset/product from being installed/removed or abort the entire session.

In addition, **swinstall** and **swremove** can execute scripts immediately before and immediately after the fileset/product has been installed or removed. These scripts usually perform additional file install or remove operations.

The **swconfig**, **swinstall**, and **swremove** commands will also execute configuration and unconfiguration scripts on an installed fileset/product to configure or unconfigure the system for the software.

The **swverify** command will execute a verification script which can analyze the configured fileset/product to verify that it is configured properly.

Software States

The SD commands transition products and filesets through a number of *states*.

During installation, software is transitioned through the following states: non-existent, TRANSIENT, INSTALLED, and CONFIGURED. During removal, software is transitioned through these states: CONFIGURED, INSTALLED, TRANSIENT, and non-existent.

When packaging or copying software into a depot, the software is transitioned through the following states: non-existent, TRANSIENT, and AVAILABLE. When removing software from a depot, the software is transitioned through these states: AVAILABLE, TRANSIENT, and non-existent.

If a task fails during any TRANSIENT state, the state is set to CORRUPT.

Session Files

Each invocation of an SD command defines a task session. Most SD commands automatically save options, source information, software selections, and target selections before the task actually commences. You can build, save, and reuse additional sessions with most commands.

Software and Target Lists

Most SD commands support software and target selections from separate input files. You can reuse files containing sets of software or target configurations as input to the commands.

Options

The following options are supported by one or more of the SD commands. Refer to the manual pages for each command for the options specific to that command.

XToolkit Options

The interactive commands support a subset of the standard X Toolkit options to control the appearance of the GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X(1)* manual page for a definition of these options.

- d** Causes the command to operate on *target_selections* which are software depots rather than root directories.
 - r** (Optional) Causes the command to operate on *target_selections* that are alternate root directories (root file systems other than /).
- Note that you cannot use this option to relocate software during installation. You must use the **l=location** syntax in the software selection component.
- i** Runs the command in interactive mode (Graphical User Interface).

(Hewlett-Packard Company)

- 1 (HP-UX 10.* only) Runs the command in *linkinstall* mode, which makes software installed under a server's **shared root** available to a diskless client's **private root**.

When run in *linkinstall* mode, **swinstall**:

- Creates NFS mounts to the software to make it accessible from the target. This may involve delayed mounting for alternate roots.
- Modifies the target's *fstab* file.
- Modifies the source's **exports** file to add mount permission for the target.

Mounts are created by examining the *share_link* product attribute. Not all products support **linkinstall**. Some products may be visible without creating a new mount if they reside under an existing one.

- p Previews the task by executing the session through the analysis phase and exiting before the command begins to perform the actual task. This option only applies to non-interactive sessions.
- R Recursively include all objects to the fileset level using **swlist**, (and to the end_target level using the HP OpenView Software Distributor **swjob** command).
- u Undo variation of the operation, unconfiguring software using **swconfig**, unregistering the specified objects using **swreg**, (or removing the specified jobs using the HP OpenView Software Distributor **swjob** command).
- v Turns on verbose output to stdout. (The command log file is not affected by this option.) By default, verbose output is enabled for all the SD commands.
- V List the SDU data model revisions that **swpackage** supports.
- a *attribute*
Specifies particular attributes to display or modify using **swlist**, **swmodify**, (or the HP OpenView Software Distributor **swjob** command).
- c *catalog*
Specifies the pathname of an exported catalog which stores copies of the response files created by request scripts.
- C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **\$HOME/.sw/sessions/**. You can recall a session file with the **-S session_file** option. From an interactive session, you can save session information into a file at any time with the *File/Save Session* or *File/Save Session As* option. You can save session information from a command-line session with the **-C session_file** option. In addition, each command automatically creates a session file of the most recent session information and names the file **\$HOME/.sw/sessions/sw<task>.last**.
- D *acl_entry*
Deletes an existing entry from the ACL associated with the specified objects using **swacl**.
- f *software_file*
Read the list of *selections* from *software_file* instead of (or in addition to) the command line operands.
- F *acl_file*
Assigns the ACL contained in *acl_file* to the specified object using **swacl**.
- J *job_id*
(HP OpenView Software Distributor only) Executes the previously scheduled job. This option is used by the **swagentd** to initiate scheduled jobs.
- l *level*
List all objects at the specified *level* when using **swlist**, or define the level of the objects when using **swacl**, or **swreg**.
- M *acl_entry*
Adds a new ACL entry or changes the permissions of an existing entry using **swacl**.
- Q *date* (HP OpenView Software Distributor only) Schedules the command for the specified date and time.

(Hewlett-Packard Company)

- s *source*
Specifies source depot, PSF file, or tape from which software will be installed, copied, listed, or packaged.
- S *session_file*
Execute **swinstall** or **swcopy** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C session_file** option.
- t *target_file*
Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line operands.
- x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.
- X *option_file*
Read the session options and behaviors from *option_file*. These values defined in this file override the default values.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "selections at targets".

Software Selections

The *selections* operands consist of *software_selections* for most SD commands. For the **swjob** (*HP Open-View Software Distributor*) and **swreg** commands, the selections can be *job_ids* and *roots_or_depots* respectively.

The SD commands support the following syntax for each *software_selection*:

bundle[*.product*][*.subproduct*][*.fileset*]][*, version*]

product[*.subproduct*][*.fileset*]][*, version*]

The **version** component has the form:

[*,r* *<op>* *revision*][*,a* *<op>* *arch*][*,v* *<op>* *vendor*]
 [*,c* *<op>* *category*][*,l=location*][*,fr* *<op>* *revision*]
 [*,fa* *<op>* *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

==, >=, <=, <, >, or !=

which performs individual comparisons on dot-separated fields.

For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches. Shell patterns are not allowed with these operators.

- The **=** (equals) relational operator lets you specify selections for **swlist**, **swpackage**, and **swmodify** with the following shell wildcard and pattern-matching notations:

[], *, ?, !

For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=A.12, r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.
- No space or tab characters are allowed in a software selection.

(Hewlett-Packard Company)

- The software *instance_id* can take the place of the version component. It has the form:

`[instance_id]`

within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The ***** software specification selects all products. It is not allowed when removing software from the root directory (/).

Target Selections

The SD commands support this syntax for each *target_selection*.

`[host][:][/ directory]`

The **:** (colon) is required if both a host and directory are specified.

The following PC information applies only to HP OpenView Software Distributor.

The **swinstall** and **swjob** commands support the following syntax for specifying PCs:

`[pc_controller][:][pc_target]`

- The PC controller is a fanout server.
- The PC target may be a PC machine, user, or group name.
- Valid targets for a PC controller can be listed using:

swlist -l machine | user | group

PC targets can be further qualified for whether they refer to a PC machine, user, or group type with the following syntax:

`name[,t=type][,k=address]`

- The *type* must be specified when a name applies to more than one of a **machine**, **user**, or **group**. (The *address* is used internally for machines and is generally not needed on the command line.)
- The keyword ***** can be substituted for **pc_target**, specifying an installation to all target machines:

@ pc_controller::*

EXTERNAL INPUTS AND INFLUENCES**Default Options**

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults the system-wide default values.

\$HOME/.swdefaults the user-specific default values.

Values must be specified in the defaults file using this syntax:

`[command_name.]option=value`

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command -x option=value

command -X option_file

The following section lists all of the keywords supported by the SD commands. The keywords that are supported for individual commands are also listed in each command's manual page. If a default value exists, it is listed after the **"=**". The commands that this option applies to are also specified.

agent=/usr/sbin/swagent

The location of the agent program invoked by the daemon.

Applies to **swagentd**.

(Hewlett-Packard Company)

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when **-p** (preview) is used. This enhances network reliability and performance. The default is **true** means the target agent automatically exits when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

Applies to **swconfig**, **swcopy**, **swinstall**, **swremove**, **swverify**.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

Applies to **swcopy**, **swinstall**, **swjob**, **swlist**, **swremove**, **swverify**.

allow_downdate=false

Prevents the installation of an older revision of fileset that already exists at the targets. (Many software products do not support "downdating".) If set to **true**, the older revision can be installed.

Applies to **swinstall**.

allow_incompatible=false

Requires that the software products which are being installed be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

Applies to **swconfig**, **swinstall**, and **swverify**.

allow_multiple_versions=false

Prevents the installation or configuration of another, independent version of a product when a version already is already installed or configured at the target.

If set to **true**, another version of an existing product can be installed into a new location, or can be configured in its new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

Applies to **swconfig**, **swinstall**, and **swverify**.

alternate_source=

Defines the alternate source which the agent will use when the **use_alternate_source** option is set to **true**. The alternate source is specified using the syntax:

`[host][:][path]`

If the host portion is not specified, then the local host is used. If the path portion is not specified, then the path sent by the command is used. The protocol sequence and endpoint given by the option **swagent.rpc_binding_info** are used when the agent attempts to contact an alternate source depot.

Applies to **swagent**.

ask=true (swask only)**ask=false (swconfig and swinstall)**

Executes a **request script**, which asks for a user response. If **ask=as_needed**, **swinstall** executes the request script only if a response file does not already exist in the control directory. See **swask(1M)** for more information on request scripts.

Applies to **swask**, **swconfig**, and **swinstall**.

(Hewlett-Packard Company)

auto_kernel_build=true

Normally set to true. Specifies whether the removal of a kernel fileset should rebuild the kernel or not. If the kernel rebuild succeeds, the system automatically reboots. If set to false, the system continues to run the current kernel.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

Applies to **swremove** only.

autoreboot=false

Prevents the installation or removal of software requiring a reboot from the non-interactive interface. If set to **true**, then software can be installed or removed, after which the target system(s) will automatically reboot.

An interactive session always asks for confirmation before software requiring a reboot is installed or removed.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

Applies to **swinstall** and **swremove**.

autorecover_product=false

Causes **swinstall** to remove the original files as they are updated. If an error occurs during the installation (e.g. network failure), then the original files are lost, and the installation must be re-tried.

If set to **true**, all files are saved as backup copies until all filesets in the current product loading are complete; then they are removed. At the cost of a temporary increase in disk space and slower performance, this allows for automatic recovery of the original filesets in that product if the load fails.

Applies only to **swinstall**.

autoremove_job=false

This option applies only to HP OpenView Software Distributor.

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or target log files) cannot be queried with **swjob**.

Install jobs to PCs can not be automatically removed. They should not be removed until the job completes on all PC targets.

autoselect_dependencies=true

Controls the automatic selection of prerequisite and corequisite software that is not explicitly selected by the user. When set to **true**, the requisite software is automatically selected for configuration. When set to **false**, requisite software which is not explicitly selected is not automatically selected for configuration.

Applies to **swconfig**, **swcopy**, **swinstall**, and **swverify**.

autoselect_dependents=false

Controls the automatic selection of dependent software that is not explicitly selected by the user. A dependent is the opposite of a requisite. A dependent fileset has established either a prerequisite or a corequisite on the fileset under discussion. Specifying **true** causes dependent software to be automatically selected for the operation. The default, **false** causes dependent software, which is not explicitly selected, to not be automatically selected for the operation.

Applies to **swconfig** and **swremove**.

(Hewlett-Packard Company)

autoselect_patches=true

Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects for a **swinstall** or **swcopy** operation. When set to **false**, the patches corresponding to the selected object will not be automatically selected.

The **patch_filter** option can be used in conjunction with **autoselect_patches**.

Applies to **swask**, **swinstall**, and **swcopy**.

autoselect_reference_bundles=true

If **true**, bundles that are **sticky** will be automatically installed, or copied, along with the software it is made up of. If **false**, the software can be installed, or copied, without automatically including **sticky** bundles that contain it.

For **swremove**, if set to **true**, any bundle with the **is_sticky** attribute set to **true** is removed automatically when the last of its contents is removed. If set to **false**, the sticky bundles will not be automatically removed.

Applies to **swcopy**, **swinstall**, and **swremove**.

check_contents=true

Causes **swverify** to verify the time stamp, size, and checksum attributes of files. If set to **false**, these attributes are not verified.

Applies to **swverify**.

check_permissions=true

Causes **swverify** to verify the mode, owner, UID, group, and GID attributes of installed files. If set to **false**, these attributes are not verified.

Applies to **swverify**.

check_requisites=true

Causes **swverify** to verify that the prerequisite and corequisite dependencies of the software selections are being met. If set to **false**, these checks are not performed.

Applies to **swverify**.

check_scripts=true

Causes **swverify** to run the fileset/product verify scripts for installed software. If set to **false**, these scripts are not executed.

Applies to **swverify**.

check_volatile=false

Causes **swverify** to not verify those files marked as volatile (i.e. can be changed). If set to **true**, volatile files are also checked (for installed software).

Applies to **swverify**.

codeword=

Provides the "codeword" needed to unlock protected HP CD-ROM software.

Some HP software products are shipped on CD-ROM as "protected" products. That is, they cannot be installed or copied unless a "codeword" and "customer ID" are provided. The codeword is found on the CD-ROM certificate which you received from HP. You may use this default specification on the command line or the SD-UX Interactive User Interface to enter the codeword.

This default stores the codeword for future reference; it needs to be entered only once. If a new HP product is purchased and a previous codeword has already been entered for that CD-ROM, just enter the new codeword as usual and the codewords will be merged internally.

compress_cmd=/usr/contrib/bin/gzip

Defines the command called to compress files before installing, copying or packaging. If the **compression_type** option is set to other than **gzip** or **compress**, this path must be changed.

Applies to **swpackage** and **swagent**.

(Hewlett-Packard Company)

compress_files=false

If set to **true**, files are compressed, if not already compressed, before transfer from a source. This will enhance performance on slower networks for **swcopy** and **swinstall**, and will result in smaller depots for **swcopy** and **swpackage**, unless the **uncompress_files** is also set to **true**.

Applies to **swcopy**, **swinstall**, and **swpackage**.

compression_type=gzip

Defines the default compression type used by the agent when it compresses files during or after transmission. If **uncompress_files** is set to **false**, the **compression_type** is recorded for each file compressed so that the correct uncompression can later be applied during a **swinstall**, or a **swcopy** with **uncompress_files** set to **true**. The **compress_cmd** specified must produce files with the **compression_type** specified. The **uncompress_cmd** must be able to process files of the **compression_type** specified unless the format is **gzip**, which is uncompressed by the internal uncompressor (**funzip**).

Applies to **swagent**.

config_cleanup_cmd=/usr/sbin/sw/config_clean

Defines the script called by the agent to perform release-specific configure cleanup steps.

Applies to **swagent**.

control_files=

When adding or deleting control file objects, this option lists the tags of those control files. There is no supplied default. If there is more than one tag, they must be separated by whitespace and surrounded by quotes.

Applies to **swmodify**.

controller_source=

Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

`[host][:][path]`

This option has no effect on which sources the target uses and is ignored when used with an Interactive User Interface.

Applies to **swcopy**, **swconfig**, **swinstall**, **swremove**, and **swverify**.

create_target_acls=true

If creating a target depot, **swpackage** will create Access Control Lists (ACLs) for the depot (if it is new) and all products being packaged into it. If set to **false**, and if the user is the superuser, **swpackage** will not create ACLs. (The **swpackage** command never creates ACLs when software is packaged on to a distribution tape.)

Applies to **swpackage**.

create_target_path=true

Causes the agent to create the target directory if it does not already exist. If set to **false**, a new target directory will not be created. This option can prevent the erroneous creation of new target depots.

Applies to **swcopy** and **swinstall**.

customer_id=

This number, also printed on the Software Certificate, is used to "unlock" protected software and restrict its installation to a specific site or owner. It is entered using the **-x customer_id=** option or by using the Interactive User Interface. The **customer_id** can be used on any HP-UX 10.0X compatible HP9000 system.

(Hewlett-Packard Company)

defer_configure=false

Causes **swinstall** to automatically configure the *software_selections* after they are installed. When an alternate root directory is specified, **swinstall** never performs the configuration task, since only hosts using the software should be configured. If set to **true**, this option allows configuration to be deferred even when the root directory is **.**

When installing a successive version of a product, it will not be configured if another version is already configured. The **swconfig** command must be run separately.

Applies to **swinstall**.

distribution_source_directory=/var/spool/sw

Defines the default distribution directory to read as the source (when the *source_type* is *directory*). The **-s** option overrides this default.

Applies to **swcopy**, **swinstall**, and **swpackage**.

distribution_target_directory=/var/spool/sw

Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

Applies to **swacl**, **swcopy**, **swlist**, **swmodify**, **swpackage**, **swreg**, **swremove**, and **swverify**.

distribution_target_serial=/dev/rmt/0m

Defines the default location of the target tape device file. The *target_selection* operand overrides this default.

Applies to **swpackage**.

enforce_dependencies=true

Requires that all dependencies specified by the *software_selections* be resolved either in the specified source, or at the *target_selections* themselves.

The **swconfig**, **swcopy**, and **swinstall** commands will not proceed unless the dependencies have also been selected or already exist at the target in the correct state (INSTALLED, CONFIGURED, or AVAILABLE). This prevents unusable software from being installed on the system. It also ensures that depots contain usable sets of software.

For **swremove**, if a selected fileset has dependents (i.e. other software depends on the fileset) and they are not selected, do not remove the selected filesets.

If set to **false**, dependencies will still be checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite dependencies, if not enforced, may cause the installation or configuration to fail.

Applies to **swconfig**, **swcopy**, **swinstall**, **swremove**, and **swverify**.

enforce_dsa=true

Prevents a command from proceeding past the analysis phase if the disk space required is beyond the available free space of the impacted file systems. If set to **false**, then the install, copy, or package operation will use the file systems' minfree space and may fail because it reaches the file system's absolute limit.

Applies to **swcopy**, **swinstall**, and **swpackage**.

enforce_kernbld_failure=true

Prevents **swinstall** from proceeding past the kernel build phase if the kernel build processes fail. If set to **false**, then the install operation will continue (without suspension if in the interactive mode) despite failure or warnings from either the system preparation process or the kernel build process.

Applies to **swinstall**.

enforce_scripts=true

If a fileset/product *checkinstall* or *checkremove* script fails (i.e. returns with exit code 1), none of the filesets in that product will be installed or removed. If set to **false**, the install or remove operation will proceed even when a check script fails.

Applies to **swinstall** and **swremove**.

S

(Hewlett-Packard Company)

files= When adding or deleting file objects, this option lists the pathnames of those file objects. There is no supplied default. If there is more than one pathname, they must be separated by whitespace.

Applies to **swmodify**.

follow_symlinks=false

Do not follow symbolic links in the package source files, but include the symbolic links in the packaged products. A value of **true** for this keyword causes **swpackage** to follow symbolic links in the package source files and include the files they reference in the packaged products.

Applies to **swpackage**.

force_job_removal=false

This option applies only to HP OpenView Software Distributor.

By default, the job information is removed from the central controller only after removing the job information stored on each of the targets succeeds. If the job should be removed regardless of the success of the removal of job information from targets, set this option to **true**.

Applies to **swjob**.

force_single_target=false

This option applies to HP-UX 10. only.*

This option applies only to the Interactive User Interface when no SD-OV license is in effect on a system that is a diskless server. It causes **swremove** to run in a single target mode, even though a diskless server normally causes **swremove** to run in multi-target mode.

include_file_revisions=false

Do not include each source file's revision attribute in the products being packaged. Because this operation is time consuming, by default the revision attributes are not included. If set to **true**, **swpackage** will execute *what*(1) and possibly *ident*(1) (in that order) to try to determine a file's revision attribute.

Applies to **swpackage**.

install_cleanup_cmd=/usr/sbin/sw/install_clean

Defines the script called by the agent to perform release-specific install cleanup steps immediately after the last postinstall script has been run. For an OS update, this script should at least remove commands that were saved by the **install_setup** script. This script is executed after all filesets have been installed, just before the reboot to the new operating system.

Applies to **swagent**.

install_setup_cmd=/usr/sbin/sw/install_setup

Defines the script called by the agent to perform release-specific install preparation. For an OS update, this script should at least copy commands needed for the checkinstall, preinstall, and postinstall scripts to a path where they can be accessed while the real commands are being updated. This script is executed before any kernel filesets are loaded.

Applies to **swagent**.

job_polling_interval=30

This option applies only to HP OpenView Software Distributor.

Defines the polling interval, in minutes, used by the daemon. It specifies how often a PC installjob will be polled in order to cache the progress of remote targets on the controller.

Applies to **swinstall**.

S

(Hewlett-Packard Company)

job_title=

This option applies only to HP OpenView Software Distributor.

This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

Applies to **swconfig**, **swcopy**, **swinstall**, **swremove**, and **swverify**.

kernel_build_cmd=/usr/sbin/mk_kernel

Defines the script called by the agent for kernel building.

Applies to **swagent**.

kernel_path=/stand/vmunix

Defines the path to the system's bootable kernel. This path is passed to the **kernel_build_cmd** via the **SW_KERNEL_PATH** environment variable.

Applies to **swagent**.

layout_version=1.0

Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8". SD for HP-UX version 10.10 and later can read or write either layout version.

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

The version used by **swpackage** can be controlled by specifying the *layout_version* attribute in the product specification file (PSF). However, if the *layout_version* attribute in the PSF is 1.0, the *is_locatable* attribute defaults to true in all cases, and must be explicitly set to false. (See *swpackage(4)* for more information on PSFs.)

Layout version 1.0 adds significant functionality not recognized by systems supporting only 0.8, including:

- Category class objects (formerly the **category** and **category_title** attributes within the bundle or product class).
- Patch-handling attributes, including **applied_patches**, **is_patch**, and **patch_state**.
- The fileset **architecture** attribute, which permits you to specify the architecture of the target system on which the product will run.

In addition to adding new attributes and objects, layout_version 1.0 changes the following preexisting 0.8 objects and attributes as follows:

- Replaces the depot **media_sequence_number** with the **media** object with a **sequence_number** attribute.
- Replaces the **vendor** definition within products and bundles with a **vendor_tag** attribute and a corresponding **vendor** object defined outside the product or bundle.
- Pluralizes the **corequisite** and **prerequisite** fileset attributes (to **corequisites** and **prerequisites**).
- Changes the **timestamp** attribute to **mod_time**.

Applies to **swpackage**, **swcopy**, **swmodify**, and **swlist**.

(Hewlett-Packard Company)

level= Specifies a software *level* for **swlist**, **swacl**, or **swreg**.

For **swlist**:

Lists all objects down to the specified *level*. Both the specified level(s) and the depth of the specified *software_selections* control the depth of the **swlist** output. The supported software levels are:

bundle	Show all objects down to the bundle level.
product	Show all objects down to the product level. Also use <i>-l bundle -l product</i> to show bundles.
subproduct	Show all objects down to the subproduct level.
fileset	Show all objects down to the fileset level. Also use <i>-l fileset -l subproduct</i> to show subproducts.
file	Show all objects down to the file level (i.e. depots, products, filesets, and files).
category	Show all categories of available software objects.
patch	Show all applied patches.

The supported depot and root levels are:

depot	Show only the depot level (i.e. depots which exist at the specified target hosts).
root	List all alternate roots.
shroot	List all registered shared roots (<i>HP-UX 10.X only</i>).
prroot	List all registered private roots (<i>HP-UX 10.X only</i>).

The machine, user, and group levels apply only to HP OpenView Software Distributor PC target files.

machine	Show the <i>machines</i> known to a PC controller.
user	Show the <i>users</i> known to a PC controller.
group	Show the <i>groups</i> known to a PC controller.

For **swacl**:

The **level** option defines the level of ACLs to view or modify:

host	View/modify the ACL protecting the host system(s) identified by the <i>target_selections</i> .
depot	View/modify the ACL protecting the software depot(s) identified by the <i>target_selections</i> .
root	View/modify the ACL protecting the root file system(s) identified by the <i>target_selections</i> .
product	View/modify the ACL protecting the software product identified by the <i>software_selection</i> . Applies only to products in depots, not installed products in roots.
product_template	View/modify the template ACL used to initialize the ACL(s) of future product(s) added to the software depot(s) identified by the <i>target_selections</i> .
global_soc_template	View/modify the template ACL used to initialize the ACL(s) of future software depot(s) or root file system(s) added to the host(s) identified by the <i>target_selections</i> .
global_product_template	View/modify the template ACL used to initialize the product_template ACL(s) of future software depot(s) added to the host(s) identified by the <i>target_selections</i> .

S

(Hewlett-Packard Company)

For **swreg**:

The **level** option defines the level of object to register or unregister:

- depot** Depots which exist at the specified target hosts.
- root** All alternate roots.
- shroot** All registered shared roots (*HP-UX 10.X only*).
- prroot** All registered private roots (*HP-UX 10.X only*).

Applies to **swacl**, **swlist**, and **swreg**.

log_msgid=0

Controls whether numeric identification numbers are prepended to logfile messages produced by SD:

- 0 (default) No identifiers are attached to messages.
- 1 Applies to ERROR messages only.
- 2 Applies to ERROR and WARNING messages.
- 3 Applies to ERROR, WARNING, and NOTE messages.
- 4 Applies to ERROR, WARNING, NOTE, and certain other logfile messages.

Applies to **swconfig**, **swcopy**, **swinstall**, **swreg**, **swremove**, and **swverify**.

logdetail=false

The **logdetail** option controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option.

Here are the possible combinations of **loglevel** and **logdetail** options:

Log Level	Log Detail	Information Included
loglevel=0		No information is written to the logfile.
loglevel=1	logdetail=false	Only key events are logged; this is the default.
loglevel=1	logdetail=true	Event detail as above plus task progress messages. Setting loglevel=1 is not necessary, it is the default.
loglevel=2	logdetail=false	Event and file level messages only. Setting the logdetail=false option is not necessary.
loglevel=2	logdetail=true	All information is logged. Setting both loglevel=2 and logdetail=true options is required. This combination may produce the same logfile behavior as previous HP-UX 10.x releases.

Applies to **swconfig**, **swcopy**, **swinstall**, **swreg**, **swremove**, and **swverify**.

logfile=/var/adm/sw/sw<command>.log

Defines the default log file for each SD command. (The agent log files are always located relative to the target depot or target root, e.g. **/var/spool/sw/swagent.log** and **/var/adm/sw/swagent.log**.)

Applies to all commands except **swacl**, **swlist**, and **swjob** (HP OpenView Software Distributor).

(Hewlett-Packard Company)

loglevel=1

Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile by prepending identification numbers to SD logfile messages. This information is in addition to the detail controlled by the **logdetail** option. See **logdetail** for more information.

A value of

- 0 provides no information to the log files.
- 1 enables verbose logging to the log files.
- 2 enables very verbose logging to the log files.

Applies to **swconfig**, **swcopy**, **swinstall**, **swmodify**, **swpackage**, **swremove**, and **swverify**.

match_target=false

If set to **true**, software selection is done by locating filesets on the source that match the target system's installed filesets. If multiple targets are specified, the first in the list is used as the basis for selections.

Applies to **swinstall**.

max_agents=-1

The maximum number of agents that are permitted to run simultaneously. The value of -1 means that there is no limit.

Applies to **swagentd**.

media_capacity=1330

If creating a distribution tape, this keyword specifies the capacity of the tape in Mbytes. This option is required if the media is not a DDS tape or a disk file. Without this option, **swpackage** sets the size to 1330 Mbytes for tape and "free space up to minfree" on a disk file.

Applies to **swpackage**.

media_type=directory

Defines the type of distribution to create. The recognized types are **directory** and **tape**.

Applies to **swpackage**.

minimum_job_polling_interval=1

This option applies only to HP OpenView Software Distributor.

Defines how often, in minutes, the daemon will wake up and scan the job queue to determine if any scheduled jobs need to be initiated or if any PC install jobs need their remote target status cached locally (see **job_polling_interval**). If set to 0, no scheduled jobs will be initiated, and no caching of PC install jobs will occur.

Applies to **swagentd**.

mount_all_filesystems=true

By default, the SD commands attempt to mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point, and that the expected files are available for a remove or verify operation.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

Applies to **swconfig**, **swcopy**, **swinstall**, **swremove**, and **swverify**.

mount_cmd=/sbin/mount

Defines the command called by the agent to mount all filesystems.

Applies to **swagent**.

(Hewlett-Packard Company)

objects_to_register=

Defines the default objects to register or unregister. There is no supplied default (see **select_local** above). If there is more than one object, they must be separated by spaces.

Applies to **swreg**.

one_liner=

Defines the attributes which are listed in the non-verbose listing.

Applies to **swlist** and HP OpenView Software Distributor's **swjob**.

os_name

This option can be used in conjunction with **os_release** to specify fileset selection for an HP-UX update. **os_name** should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

```
swlist -d -a readme SW-DIST [@ host:/depot ]
```

Applies to **swinstall**.

os_release

This option can be used in conjunction with **os_name** to specify fileset selection for an HP-UX update. **os_release** should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

```
swlist -d -a readme SW-DIST [@ host:/depot ]
```

Applies to **swinstall**.

package_in_place=false

If set to **true**, **swpackage** will package the specified products such that the target depot will not contain the files that make up a product. Instead, **swpackage** inserts references to the original source files used to build a product. This behavior allows products to be packaged without consuming the full disk space of copying all the source files into the target depot.

Applies to **swpackage**.

patch_commit=false

Commits a patch by removing files saved for patch rollback. When set to **true**, and run with **swmodify**, you cannot roll back (remove) a patch unless you remove the associated base software that the patch modified.

Applies to **swmodify**.

patch_filter=*. *

Specifies a *software_specification* for a patch filter. The default value is ***. ***.

This option can be used in conjunction with the **autoselect_patches** and **patch_match_target** options to filter the selected patches to meet the criteria specified by *software_specification*.

Applies to **swask**, **swcopy**, and **swinstall**.

patch_match_target=false

If set to **true**, this option selects the latest patches (software identified by the *is_patch* attribute) that correspond to software on the target root or depot.

The **patch_filter=** option can be used in conjunction with **patch_match_target**.

Applies to **swcopy** and **swinstall**.

patch_one_liner=title patch_state

Specifies the attributes displayed for each object listed when the **-l patch** option is invoked and when no **-a** or **-v** option is specified. The default display attributes are **title** and **patch_state**.

Applies to **swlist** and **swjob**.

(Hewlett-Packard Company)

patch_save_files=true

Saves patched files, which permits future rollback of patches. When set to **false**, patches cannot be rolled back (removed) unless the base software modified by the patch is removed at the same time.

Applies to **swinstall**.

poll_now=false

This option applies only to HP OpenView Software Distributor.

The status information displayed for a PC install job is as recent as the last time the daemon polled remote targets for information (see the option **job_polling_interval**). If the most recent status is wanted set this option to **true**.

Applies to **swjob**.

polling_interval=2

Defines in seconds the polling interval used by interactive (GUI) sessions. It specifies how often each target agent will be polled to obtain status information about the task being performed. When operating across wide-area networks, the polling interval can be increased to reduce network overhead.

Applies to **swcopy**, **swinstall**, and **swremove**.

reboot_cmd=/sbin/reboot

Defines the command called by the agent to reboot the system.

Applies to **swagent**.

reconfigure=false

Prevents software which is already in the CONFIGURED state from being reconfigured. If set to **true**, CONFIGURED software can be reconfigured.

Applies to **swconfig**.

register_new_depot=true

Causes **swcopy** to register a newly created depot with the local **swagentd**. This action allows other SD commands to automatically "see" this depot. If set to **false**, a new depot will not be automatically registered. (It can be registered later with the **swreg** command.)

Applies to **swcopy**.

register_new_root=true

Causes **swinstall** to register a newly created alternate root with the local **swagentd**. This action allows other SD commands to automatically "see" this root. If set to **false**, a new root will not be automatically registered. (It can be registered later with the **swreg** command.)

Applies to **swinstall**.

reinstall=false

When re-installing (or re-copying) an existing version of a fileset, this option causes that fileset to be skipped, i.e. not re-installed. If set to **true**, the fileset will be re-installed (re-copied).

Applies to **swinstall** and **swcopy**.

reinstall_files=true

Causes all the files in a fileset to always be re-installed, re-copied, or re-packaged, even when the file already exists at the target and is identical to the new file. If set to **false**, files that have the same *checksum* (see next option), size and time stamp will not be re-installed, re-copied, or re-packaged. This check enhances performance on slow networks or slow disks.

Applies to **swinstall**, **swcopy**, and **swpackage**.

(Hewlett-Packard Company)

reinstall_files_use_cksum=true

This option affects the operation when the **reinstall_files** option is set to false. It causes the checksums of the new and old file to be computed and compared to determine if the new file should replace the old one. (The checksum is slower, but is a more robust way to check for files being equivalent.) If set to **false**, the checksums are not computed, and files are (not) reinstalled based only on their size and time stamp. For **swpackage**, the default value for this option is **false**.

Applies to **swcopy**, **swinstall**, and **swpackage**.

remove_empty_depot=true

Remove an empty depot when the last product is removed. If set to **false**, an empty depot will not be removed, preserving any depot ACLs.

Applies to **swremove**.

remove_fanout_depot=true

This option applies only to HP OpenView Software Distributor.

When an install job to a PC is removed the software associated with that job is automatically removed from the PC depot. If the software that is part of this job is the same software being used by another job, then be sure to not delete the software as part of the job removal. If the software on the PC depot should be retained, set this option to **false**.

Applies to **swjob**.

remove_obsolete_filesets=false

Controls whether **swcopy** automatically removes obsolete filesets from target products in the target depot. If set to **true**, **swcopy** removes obsolete filesets from the target products that were written to during the copy process. Removal occurs after the copy is complete. Filesets are defined as obsolete if they were not part of the most recent packaging of the product residing on the source depot.

Applies to **swcopy**.

remove_setup_cmd=/usr/sbin/sw/remove_setup

Defines the script called by the agent to perform release-specific removal preparation. For an OS update, this script invokes the **mlink** command when a fileset is removed.

Applies to **swagentd**.

retry_rpc=1

Defines the number of times a lost source connection will be retried during file transfers. A lost connection is one that has timed out. When used in conjunction with the **rpc_timeout** option, the success of installing over slow or busy networks can be increased. If set to zero, then any **rpc_timeout** to the source will cause the task to abort. If set from 1 to 9, then the install of each fileset will be attempted that number of times. The **reinstall_files** option should also be set to false to avoid installing files within the fileset that were successfully installed.

Applies to **swcopy** and **swinstall**.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and on which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

The value (or values for **swagentd**) can have following form:

- A DCE string binding containing a protocol sequence and an endpoint. The syntax is: **protocol_sequence:[endpoint]**.
- The name of a DCE protocol sequence with no endpoint specified. The syntax is: **protocol_sequence**, for example **ncadg_ip_udp** or **ncacn_ip_tcp**. (A trailing **:** can be attached to the protocol sequence, it has no effect.) Since no endpoint is specified, the DCE endpoint mapper **rpcd** must be running and will be used to find the endpoint registered by the **swagentd**.
- The literal string **all**. This entry means to use (try) all protocol sequences supported by the DCE RPC. It should be the only entry in the list. The DCE endpoint mapper **rpcd** also must be running in order to use this option.

Applies to all commands except **swask**, **swpackage**, and **swmodify**.

rpc_binding_info_alt_source=ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) used when the agent attempts to contact an alternate source depot specified by the **alternate_source** option. HP-UX supports both the **udp(ncadg_ip_udp:[2121])** and **tcp(ncacn_ip_tcp:[2121])** protocol sequence/endpoint. SD on SunOS only supports **udp(ncadg_ip_udp:[2121])**. By default **udp** is used.

Applies to **swagentd**.

rpc_timeout=5

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

Applies to all commands except **swpackage** and **swmodify**.

select_local=true

If no **target_selections** are specified, select the default **target_directory** of the local host as the **target_selection** for the command.

Applies to **swacl**, **swconfig**, **swcopy**, **swinstall**, **swlist**, **swreg**, **swremove**, and **swverify**.

software=

Defines the default **software_selections**. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

Applies to all commands except **swreg** and HP OpenView Software Distributor's **swjob**.

software_view=products

Indicates the software view to be used by the interactive interface of the commands and by **swlist** for the default listing level. It can be set to **products**, **all_bundles**, or a bundle category tag (to indicate to show only bundles of that category).

Applies to **swcopy**, **swinstall**, **swlist**, and **swremove**.

source_cdrom=/SD_CDROM

Defines the default location of the source CD-ROM. This syntax can be **host:path**.

Applies to **swinstall**.

(Hewlett-Packard Company)

source_depot_audit=true

If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can set this option to track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. (Note that a user running **swinstall/swcopy** from a target machine cannot set this option; only the administrator of the source depot machine can set it.)

When **source_depot_audit** is set to **true**, a **swaudit.log** file is created on the source depot (for writable directory depots) or in **/var/tmp** (for *tar* images, CD-ROMs, or other non-writable depots).

Users can invoke the **swlist** interactive user interface (using **swlist -i -d**) to view, print, or save the audit information on a remote or local depot. Users can view audit information based on language preference, as long as the system has the corresponding SD message catalog files on it. For example, a user can view the source audit information in Japanese during one invocation of **swlist**, then view the same information in English at the next invocation.

Applies to **swagent**, **swinstall**, and **swlist**.

source_file=psf

Defines the default location of the source product specification file (PSF). The **host:path** syntax is not allowed, only a valid **path** can be specified. The **-s** option overrides this value.

Applies to **swpackage** and **swmodify**.

source_tape=/dev/rmt/0m

Defines the default location of the source tape, usually the character-special file of a local tape device. If the **host:path** syntax is used, the host must match the local host. The **-s** option overrides this value.

Applies to **swcopy** and **swinstall**.

source_type=directory

Defines the default source type: **cdrom**, **file**, **directory**, or **tape**. The source type derived from the **-s** option overrides this value.

Applies to **swcopy**, **swinstall**, and **swpackage**. (The values **cdrom**, and **tape** apply to **swcopy** and **swinstall** only. The value **file** applies to **swpackage** only.)

system_file_path=/stand/system

Defines the path to the kernel's template file. This path is passed to the **system_prep_cmd** via the **SW_SYSTEM_FILE_PATH** environment variable.

Applies to **swagent**.

system_prep_cmd=/usr/sbin/sysadm/system_prep

Defines the kernel build preparation script called by the agent. This script must do any necessary preparation so that control scripts can correctly configure the kernel about to be built. This script is called before any kernel filesets have been loaded.

Applies to **swagent**.

targets=

Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces. Targets are usually specified in a target input file, as operands on the command line, or in the GUI.

Applies to all commands.

uncompress_cmd=

Defines the command to uncompress files when installing, copying, or packaging. This command processes files which were stored on the media in a compressed format. If the **compression_type** of the file is **gzip** then the internal uncompression (**funzip**) is used instead of the external **uncompress_cmd**.

Applies to **swpackage** and **swagent**.

(Hewlett-Packard Company)

uncompress_files=false

If the files being transferred from a source are compressed, setting this option will uncompress the files before storing them on the target depot.

Applies to **swcopy** and **swpackage**.

use_alternate_source=false

Empowers each target agent to use its own, configured alternate source, instead of the one specified by the user. If **false**, each target agent will use the same source, namely the source specified by the user and validated by the command. If **true**, each target agent will instead use its own configured value for the source.

Applies to **swcopy** and **swinstall**.

verbose=

Controls the verbosity of a non-interactive command's output:

- 0 disables output to stdout. (Error and warning messages are always written to stderr).
- 1 enables verbose messaging to stdout.
- 2 for **swpackage** and **swmodify**, enables very verbose messaging to stdout.

For the **swlist** command, a verbose listing includes all attributes that have been defined for the appropriate level of each *software_selection* operand. The attributes are listed, one per line, prefaced by the attribute keyword.

The **-v** option overrides this default if it is set to 0.

Applies to all commands.

write_remote_files=false

Prevents the installation, copying, or packaging of files to a target which exists on a remote (NFS) file system. Also prevents the removal of files from a remote file system. All files destined for (or already on) a remote file system will be skipped.

If set to **true** and if the superuser has write permission on the remote file system, the remote files will not be skipped, but will be installed, copied, packaged, or removed.

Applies to **swcopy**, **swinstall**, **swpackage**, and **swremove**.

Session Files

Each invocation of an SD command defines a task session. Most SD commands automatically save options, source information, software selections, and target selections before the task actually commences. This lets you re-execute the command even if the session ends before the task is complete. You can also save session information from interactive or command-line sessions.

Session information is saved to the file **\$HOME/.sw/sessions/command_name.last**. This file is overwritten by each invocation of the command. The file uses the same syntax as the defaults files.

From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu.

From a command-line session, you can save session information by executing the command with the **-C session_file** option. You can specify an absolute path for a session file. If you do not specify a directory, the default location is **\$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu.

To re-execute a session from a command-line, specify the session file as the argument for the **-S** option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command-line options and parameters take precedence over the values in the session file.

(Hewlett-Packard Company)

Software and Target Lists

Most SD commands support software and target selections from separate input files (see the **-f** and **-t** options). Software and targets specified in these files will be selected for operation.

Additionally, commands that support an interactive interface read a list of possible hosts to operate on from the values found in:

```
/var/adm/sw/defaults.hosts    the system-wide default list of hosts,
$HOME/.sw/defaults.hosts     the user-specific default list of hosts.
```

Hosts in this file are not marked for operation, but provide a default list from which to choose. For each interactive command, target hosts containing roots, containing depots, and hosts serving as fanout servers are specified in separate lists (**hosts**, **hosts_with_depots**, and **fanout_servers** respectively). The list of hosts are enclosed in {} braces and separated by white space (blank, tab and newline). For example:

```
swinstall.hosts={hostA hostB hostC hostD
hostE hostF}
swinstall.fanout_servers={pc1 pc2} (HP OpenView Software Distributor only)
swcopy.hosts_with_depots={hosts}
swcopy.fanout_servers={pc1 pc2} (HP OpenView Software Distributor only)
swremove.hosts={hostA hostB hostC hostD
hostE hostF}
swremove.hosts_with_depots={hosts}
```

Most SD commands support patch filtering with the **-x patch_filter=software_specification** option. In addition, the interactive user interface commands, **swinstall** and **swcopy** read a list of possible patch filters. The user can use the values from this list for selection criteria. The lists are stored in:

```
/var/adm/sw/defaults.patchfilters
the system-wide default list of patch filters.

$HOME/.sw/defaults.patchfilters
the user-specific default list of patch filters.
```

Filters in this file are not marked for selection use but provide a default list from which the user can choose. The list of patch filters is enclosed in braces {} and separated by white space (blank, tab, or newline). For example:

```
swinstall.patch_filter_choices={
*.*,c=enhancement
*.*,c=critical
}
swremove.patch_filter_choices={
Product.Fileset,c=halts_system
}
```

The following PC information applies to HP OpenView Software Distributor only.

For installing to PCs, PC target lists are generated automatically by querying the PC file server associated with a PC controller. Any user, group, or machine known to the file server will be included in the default list from which to choose. Additionally, all machines known to the file server will by default be selected for installation when selecting a PC controller.

Environment Variables

SD programs are affected by external environment variables, set environment variables for use by the control scripts, and set additional environment variables that affect scripts run by **swinstall** and **swremove**.

External environment variables that affect the SD commands:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang(5)* for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable `script, /etc/rc.config.d/LANG`. For example, `/etc/rc.config.d/LANG`, must be set to `LANG=ja_JP.SJIS` or `LANG=ja_JP.eucJP` to make the agent and daemon log messages display in Japanese.

This variable applies to all SD commands except **swgettools**.

Environment variables that affect scripts:

SW_CONTROL_DIRECTORY

Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_LOCATION

Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH

A **PATH** variable which defines a minimum set of commands available to for use in a control script (e.g. `/sbin:/usr/bin`).

SW_ROOT_DIRECTORY

Defines the root directory in which the session is operating, either `"/` or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY** is `"/`.

SW_SESSION_OPTIONS

Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a *request* script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC

This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Additional environment variables that affect scripts run by **swinstall** and **swremove**:

SW_DEFERRED_KERNBLD

Only applies to **swinstall**. This variable is normally unset. If it is set, the actions necessary for preparing the system file `/stand/system` cannot be accomplished from within the *postinstall* scripts, but instead must be accomplished by the *configurescripts*. This occurs whenever software is installed to a directory other than `/`, such as for a cluster client system. This variable should be read only by the *configure* and *postinstall* scripts of a kernel fileset. The **swinstall** command sets these environment variables for use by the kernel preparation and build scripts.

SW_INITIAL_INSTALL

Only applies to **swinstall**. This variable is normally unset. If it is set, the **swinstall** session is being run as the back end of an initial system software installation ("cold" install).

SW_KERNEL_PATH

Only applies to **swinstall**. The path to the kernel. The default value is `/stand/vmunix`,

(Hewlett-Packard Company)

defined by the **swagent** option or **kernel_path**.

SW_SESSION_IS_KERNEL

Indicates whether a kernel build is scheduled for the current install/remove session. A **TRUE** value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to **/stand/system** are required. A null value indicates that a kernel build is not scheduled and that changes to **/stand/system** are not required.

The value of this variable is always equal to the value of **SW_SESSION_IS_REBOOT**.

SW_SESSION_IS_REBOOT

Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the values of this variables is always equal to the value of **SW_SESSION_IS_KERNEL**.

SW_SYSTEM_FILE_PATH

Only applies to **swinstall**. The path to the kernel's system file. The default value is **/stand/system**.

Signals

The SD commands catch the signals SIGQUIT and SIGINT. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Locking

SD commands use a common locking mechanism for reading and modifying both root directories and software depots. This mechanism allows multiple readers but only one writer on a root or depot.

The SD commands which modify software in an (alternate) root directory are restricted from simultaneous modification using *fcntl(2)* locking on the file

var/adm/sw/products/swlock

relative to the root directory (e.g. **/var/adm/sw/products/swlock**).

The SD commands which modify software in a depot are restricted from simultaneous modification using *fcntl(2)* locking on the file

catalog/swlock

relative to the depot directory (e.g. **/var/spool/sw/catalog/swlock**).

All commands set *fcntl(2)* read locks on roots and depots using the **swlock** file mentioned above. When a read lock is set, it prevents other SD commands from performing modifications (i.e. from setting write locks).

RETURN VALUES

Each SD command invocation returns:

- 0 The sw<task> successfully completed.
- 1 The sw<task> failed on all *target_selections*.
- 2 The sw<task> failed on some *target_selections*.

DIAGNOSTICS

The **swconfig**, **swcopy**, **swinstall**, **swmodify**, **swpackage**, **swremove**, and **swverify** commands support a preview mode, where operation will proceed through the analysis of each *target_selection*, then exit before the actual task is performed.

The HP OpenView Software Distributor interactive interface (invoked using the **sd** command) and the **swjob** command can be used to view the current status of any job as well as the controller and target log files.

Preview is only applicable for non-interactive operation, since the interactive commands wait for confirmation after analysis. In the interactive mode, you can resolve invalid conditions that the commands discover before they actually begin loading or removing files.

Standard Output

When non-interactive, the commands write messages for significant events. These events include:

- a begin and end task message,
- a message for starting the task on each host, and
- a message for completing the task on each host.

When the **verbose** option is set, summary messages about the task are also sent to the standard output.

Standard Error

When non-interactive, the commands also write messages for the following significant error events:

- a message for each host failing analysis and
- a message for each host failing the actual task.

Logging

All commands log major events on the host where the command was invoked. They log detailed events to the **swagent** log associated with each *target_selection*.

Command Log

The commands log messages to **/var/adm/sw/sw<task>.log**. (The user can specify a different logfile by modifying the **logfile** option.)

Target Log

A **swagent** process performs the actual **swacl**, **swconfig**, **swcopy**, **swinstall**, **swremove**, and **swverify** operation at each *target_selection*. For operations on target root objects, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). For operations on target depot objects, the **swagent** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**).

The **swagentd** running on a host logs events to the file **/var/adm/sw/swagentd.log**.

Source Depot Audit Log

If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. Refer to the **source_depot_audit** option in **swagent(1M)** for more information.

The following PC information applies to HP OpenView Software Distributor only.

On a PC controller, **SWAGENTD.EXE** logs events and messages to the file **... \SD\DATA\SWAGENTD.LOG**.

Command, agent, and target log files can be viewed using the **swjob** command.

LIMITATIONS

The following PC information applies to HP OpenView Software Distributor only.

For PCs, the SD commands generally only apply to the PC controller, or the PC depot on the PC controller. The **swinstall** and **swjob** commands indirectly install to and retrieve information from PC targets.

The **swpackage** command is not used for PC software. PC software is packaged using the PC Console on the PC controller, then copied (with **swcopy**) to a UNIX depot for subsequent distribution.

FILES

/dev/rmt/0m
Default source tape location.

/etc/fstab
List of volumes that should be mounted.

\$HOME/.swdefaults
Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **\$HOME/.sw/defaults**.

\$HOME/.sw/defaults.hosts
Contains the user-specific default list of hosts to manage.

\$HOME/.sw/defaults.patchfilters
Contains the user-specific default list of patch filters.

\$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/sbin/swagent
The SD agent.

/usr/lib/nls/\$LANG/sw*.cat
The SD message catalogs.

/usr/lib/sw/help/
The directory which contains the help files used by the SD GUIs' on-line help facility.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/usr/lib/sw/ui/
The directory which contains the description files used by the SD Graphical User Interfaces (GUIs).

/usr/newconfig/var/adm/sw/
The directory containing the configurable data shipped for the SD product, which is conditionally copied into **/var/adm/sw/** based on the existing configuration.

/usr/sbin/sw*
The SD commands.

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of log files.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/defaults.hosts
Contains the system-wide default list of hosts to manage.

/var/adm/sw/defaults.patchfilters
Contains the system-wide default list of patch filters.

/var/adm/sw/getdate.template
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/host_object
The file which stores the list of depots registered at the local host.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/queue/
The directory which contains the information about all active and complete install, remove, and other jobs initiated by the SD commands. Applies to HP OpenView Software Distributor only..

/var/adm/sw/security/
The directory which contains ACLs for the system itself, template ACLS, and the secrets file used to authenticate remote requests.

S

(Hewlett-Packard Company)

/var/spool/sw/

The default location of a source and target software depot.

The following applies to HP OpenView Software Distributor only.

/usr/lib/sw/examples/

The directory containing an example depot and example swpackage data.

/usr/OV/help/\$LANG/sd/

The directory which contains the help files used by the SD integration into OpenView.

/usr/OV/registration/\$LANG/sd.reg

The registration file which integrates SD into the HP OpenView Network Node Manager.

PC FILES

The following applies to HP OpenView Software Distributor only.

... \SD\AGENTS*.EXE

The SD PC agents.

... \SD\CONSOLE*.EXE

The SD PC commands.

... \SD\DATA

The directory which contains all of the configurable and non-configurable data for SD.

... \SD\DATA\DEPOT

The default location of the source and target PC depot.

... \SD\DATA\NADMIN.DST

The database which defines all PC distribution jobs and PC target status.

... \SD\DATA\NADMIN.INV

The database which defines all PC target machines available for fanout installation.

... \SD\DATA\QUEUE

The directory which contains the information about all active and complete install, remove, and other jobs initiated by the SD commands.

... \SD\DATA\SECURITY

The directory which contains ACLs for the system itself, template ACLS, and the secrets file used to authenticate remote requests.

... \SD\DATA\SWAGENTD.LOG

The log of all actions and events performed by the PC controller.

<WINDOWS> \NADMIN.INI

Contains the configurable options for the SD PC console.

<WINDOWS> \SWAGENTD.INI

Contains the configurable options for the SD PC controller.

AUTHOR

Software Distributor was developed by the Hewlett-Packard Company. **swagent**, **swcopy**, **swinstall**, **swlist**, and **swpackage** were developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax(1)*).

SEE ALSO

The *Managing HP-UX Software with SD-UX* manual, the *HP OpenView Software Distributor Administrator's Guide*, **sd(4)**, **swacl(1M)**, **swagentd(1M)**, **swask(1M)**, **swconfig(1M)**, **swcopy(1M)**, **swgettools(1M)**, **swinstall(1M)**, **swjob(1M)**, **swlist(1M)**, **swmodify(1M)**, **swpackage(1M)**, **swpackage(4)**, **swreg(1M)**, **swremove(1M)**, **swverify(1M)**.

NAME

signal: signal.h - description of signals

SYNOPSIS

```
#include <signal.h>
```

DESCRIPTION

The **<signal.h>** header defines the following symbolic constants, each of which expands to a distinct constant expression of the type:

```
void (*)(int)
```

whose value matches no declarable function.

```
SIG_DFL      Request for default signal handling.
SIG_ERR      Return value from signal() in case of error.
SIG_HOLD     Request that signal be held.
SIG_IGN      Request that signal be ignored.
```

The following data types are defined through **typedef**:

```
sig_atomic_t  Integral type of an object that can be accessed as an atomic entity, even in the presence of asynchronous interrupts
sigset_t      Integral or structure type of an object used to represent sets of signals.
pid_t         As described in <sys/types.h>.
```

This header also declares the constants that are used to refer to the signals that occur in the system. Signals defined here begin with the letters **SIG**. Each of the signals have distinct positive integral values. The value 0 is reserved for use as the null signal (see *kill(2)*). Additional implementation-dependent signals may occur in the system.

The following signals are supported on all implementations (default actions are explained below the table):

Signal	Default Action	Description
SIGABRT	ii	Process abort signal.
SIGALRM	i	Alarm clock.
SIGFPE	ii	Erroneous arithmetic operation.
SIGHUP	i	Hangup.
SIGILL	ii	Illegal instruction.
SIGINT	i	Terminal interrupt signal.
SIGKILL	i	Kill (cannot be caught or ignored).
SIGPIPE	i	Write on a pipe with no one to read it.
SIGQUIT	ii	Terminal quit signal.
SIGSEGV	ii	Invalid memory reference.
SIGTERM	i	Termination signal.
SIGUSR1	i	User-defined signal 1.
SIGUSR2	i	User-defined signal 2.
SIGCHLD	iii	Child process terminated or stopped.
SIGCONT	v	Continue executing, if stopped.
SIGSTOP	iv	Stop executing (cannot be caught or ignored).
SIGTSTP	iv	Terminal stop signal.
SIGTTIN	iv	Background process attempting read.
SIGTTOU	iv	Background process attempting write.
SIGBUS	ii	Bus error.
SIGPOLL	i	Pollable event.
SIGPROF	i	Profiling timer expired.
SIGSYS	ii	Bad system call.
SIGTRAP	ii	Trace/breakpoint trap.
SIGURG	i	High bandwidth data is available at a socket.
SIGVTALRM	i	Virtual timer expired.
SIGXCPU	ii	CPU time limit exceeded.
SIGXFSZ	ii	File size limit exceeded.

SIGRTMIN	i	First realtime signal.
SIGRTMAX	i	Last realtime signal.

The macros **SIGRTMIN** and **SIGRTMAX** evaluate to integral expressions, and specify a range that includes at least **{RTSIG_MAX}** signal numbers that are reserved for application use and for which the realtime signal extensions are supported (see *sigaction(2)*).

The default actions are as follows:

- i Abnormal termination of the process. The process is terminated with all the consequences of **_exit()** except that the status is made available to **wait()** and **waitpid()** indicates abnormal termination by the specified signal.
- ii Abnormal termination of the process. Additionally, implementation-dependent abnormal termination actions, such as creation of a core file, may occur.
- iii Ignore the signal.
- iv Stop the process.
- v Continue the process, if it is stopped; otherwise ignore the signal.

The header provides a declaration of struct **sigaction**, including at least the following members:

void	(*sa_handler)(int)	what to do on receipt of signal
sigset_t	sa_mask	set of signals to be blocked during execution of the signal handling function
int	sa_flags	special flags
void (*) (int, siginfo_t *, void *)	sa_sigaction	pointer to signal handler function

The storage occupied by *sa_handler* and *sa_sigaction* may overlap, and a portable program must not use both simultaneously.

The following are declared as constants:

SA_NOCLDSTOP	Do not generate SIGCHLD when children stop.
SIG_BLOCK	The resulting set is the union of the current set and the signal set pointed to by the argument <i>set</i> .
SIG_UNBLOCK	The resulting set is the intersection of the current set and the complement of the signal set pointed to by the argument <i>set</i> .
SIG_SETMASK	The resulting set is the signal set pointed to by the argument <i>set</i> . SA_ONSTACK Causes signal delivery to occur on an alternate stack.
SA_RESETHAND	Causes signal dispositions to be set to SIG_DFL on entry to signal handlers.
SA_RESTART	Causes certain functions to become restartable.
SA_SIGINFO	Causes extra information to be passed to signal handlers at the time of receipt of a signal.
SA_NOCLDWAIT	Causes implementations not to create zombie processes on child death.
SA_NODEFER	Causes signal not to be automatically blocked on entry to signal handler.
SS_ONSTACK	Process is executing on an alternate signal stack.
SS_DISABLE	Alternate signal stack is disabled.
MINSIGSTKSZ	Minimum stack size for a signal handler.
SIGSTKSZ	Default size in bytes for the alternate signal stack.

The **ucontext_t** structure is defined through **typedef** as described in **<ucontext.h>**.

The `<signal.h>` header defines the `stack_t` type as a structure that includes at least the following members:

```
void    *ss_sp      stack base or pointer
size_t  ss_size     stack size
int     ss_flags    flags
```

The `<signal.h>` header defines the `sigstack` structure that includes at least the following members:

```
int     ss_onstack  non-zero when signal stack is in use
void    *ss_sp      signal stack pointer
```

The `<signal.h>` header defines the `sigevent` structure that includes at least the following members:

```
int          sigev_notify  Notification type
int          sigev_signo   Signal number
union signal sigev_value   Signal value.
```

The `sigev_notify` member specifies the notification mechanism to use when an asynchronous event occurs. The following values are defined for the `sigev_notify` member:

```
SIGEV_NONE      No asynchronous notification will be delivered when the event of interest
                 occurs.

SIGEV_SIGNAL     The signal specified in sigev_signo will be generated for the process when
                 the event of interest occurs. If SA_SIGINFO is set for that signal number,
                 then the signal will be queued to the process, and the value specified in
                 sigev_value will be the sigev_value component of the generated signal. If
                 SA_SIGINFO is not set for that signal number, it is unspecified whether
                 the signal is queued, and what value, if any, is sent.
```

The `sigev_signo` member specifies the signal to be generated. The `sigev_value` member is the application-defined value to be passed to the signal-catching function at the time of the signal delivery or to be returned at signal acceptance as the `si_value` member of the `siginfo_t` structure.

The `<signal.h>` header defines `sigval` as a union that includes at least the following members:

```
int      sival_int   Integer signal value
void *   sival_ptr   Pointer signal value.
```

The `<signal.h>` header defines the `siginfo_t` type as a structure that includes at least the following members:

```
int          si_signo   signal number
int          si_errno   if non-zero, an errno value
                 associated with this signal, as
                 defined in <errno.h>

int          si_code     signal code
union signal si_value    signal value
id_t         si_pid      sending process ID
uid_t        si_uid      real user ID of sending process
void         *si_addr    address of faulting instruction
int          si_status    exit value or signal
long         si_band     band event for SIGPOLL
```

The `si_code` member contains a code identifying the cause of the signal. The following values are defined for `si_code`:

```
SI_USER      The signal was sent by kill(). The si_code may be set to SI_USER also
                 if the signal was sent by raise() or similar functions that are provided
                 as implementation extensions of kill().

SI_QUEUE     The signal was sent by sigqueue().

SI_TIMER     The signal was generated by the expiration of a timer set by
                 timer_settime().

SI_ASYNCIO   The signal was generated by the completion of an asynchronous I/O
                 request.

SI_MSGQ      The signal was generated by the arrival of a message on an empty message
                 queue.
```

If the signal was not generated by one of the functions or events listed above, the *si_code* will be set to an implementation-defined value (see below) that is not equal to any of the values defined above.

If *si_code* is one of `SI_QUEUE`, `SI_TIMER`, `SI_ASYNCIO`, or `SI_MESGQ`, then *si_value* will contain the application-specified signal value. Otherwise, the contents of *si_value* are undefined.

The macros specified in the Code column of the following table are defined for use as values of *si_code* that are signal-specific reasons why the signal was generated.

Signal	Code	Reason
SIGILL	<code>ILL_ILLOPC</code>	illegal opcode
	<code>ILL_ILLOPN</code>	illegal operand
	<code>ILL_ILLADR</code>	illegal addressing mode
	<code>ILL_ILLTRP</code>	illegal trap
	<code>ILL_PRVOPC</code>	privileged opcode
	<code>ILL_PRVREG</code>	privileged register
	<code>ILL_COPROC</code>	coprocessor error
	<code>ILL_BADSTK</code>	internal stack error
SIGFPE	<code>FPE_INTDIV</code>	integer divide by zero
	<code>FPE_INTOVF</code>	integer overflow
	<code>FPE_FLTDIV</code>	floating point divide by zero
	<code>FPE_FLTOVF</code>	floating point overflow
	<code>FPE_FLTUND</code>	floating point underflow
	<code>FPE_FLTRES</code>	floating point inexact result
	<code>FPE_FLTINV</code>	invalid floating point operation
	<code>FPE_FLTSUB</code>	subscript out of range
SIGSEGV	<code>SEGV_MAPERR</code>	address not mapped to object
	<code>SEGV_ACCERR</code>	invalid permissions for mapped object
SIGBUS	<code>BUS_ADRALN</code>	invalid address alignment
	<code>BUS_ADRERR</code>	non-existent physical address
	<code>BUS_OBJERR</code>	object specific hardware error
SIGTRAP	<code>TRAP_BRKPT</code>	process breakpoint
	<code>TRAP_TRACE</code>	process trace trap
SIGCHLD	<code>CLD_EXITED</code>	child has exited
	<code>CLD_KILLED</code>	child has terminated abnormally and did not create a core file
	<code>CLD_DUMPED</code>	child has terminated and created a core file
	<code>CLD_KILLED</code>	child was killed
	<code>CLD_DUMPED</code>	child has terminated abnormally
	<code>CLD_TRAPPED</code>	traced child has trapped
	<code>CLD_STOPPED</code>	child has stopped
	<code>CLD_CONTINUED</code>	stopped child has continued
SIGPOLL	<code>POLL_IN</code>	data input available
	<code>POLL_OUT</code>	output buffers available
	<code>POLL_MSG</code>	input message available
	<code>POLL_ERR</code>	I/O error
	<code>POLL_PRI</code>	high priority input available
	<code>POLL_HUP</code>	device disconnected

Implementations may support additional *si_code* values not included in this list, may generate values included in this list under circumstances other than those described in this list, and may contain extensions or limitations that prevent some values from being generated. Implementations will not generate a different value from the ones described in this list for circumstances described in this list.

In addition, the following signal-specific information will be available:

Signal	Member	Value
--------	--------	-------

SIGILL	void * si_addr	address of faulting instruction
SIGFPE		
SIGSEGV	void * si_addr	address of faulting memory reference
SIGBUS		
SIGCHLD	pid_t si_pid	child process ID
	int si_status	exit value or signal
	uid_t si_uid	real user ID of the process that sent the signal
SIGPOLL	long si_band	band event for POLL_IN, POLL_OUT, or POLL_MSG

For some implementations, the value of *si_addr* may be inaccurate.

The following are declared as functions and may also be defined as macros:

```
void (*bsd_signal(int sig, void (*func)(int)))(int);
int kill(pid_t pid, int sig);

int killpg(pid_t pgrp, int sig);
int raise(int sig);
int sigaction(int sig, const struct sigaction
              *act, struct sigaction *oact);
int sigaddset(sigset_t *set, int signo);

int sigaltstack(const stack_t *ss, stack_t *oss);
int sigdelset(sigset_t *set, int signo);
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);

int sighold(int sig);
int sigignore(int sig);
int siginterrupt(int sig, int flag);
int sigismember(const sigset_t *set, int signo);

int sigmask(int signal);
void (*signal(int sig, void (*func)(int)))(int);

int sigpause(int sig);
int sigpending(sigset_t *set);
int sigprocmask(int how, const sigset_t *set, sigset_t *oset);
int sigqueue(pid_t pid, int sig, const union sigval value);
int sigrlse(int sig);
void *sigset(int sig, void (*disp)(int))(int);
int sigstack(struct sigstack *ss,
             struct sigstack *oss);

int sigsuspend(const sigset_t *sigmask);
int sigtimedwait(const sigset_t *set, siginfo_t * info,
                const struct timespec *timeout);
int sigwait(const sigset_t *set, int *sig);
int sigwaitinfo(const sigset_t *set, siginfo_t * info);
```

APPLICATION USAGE

Threads Considerations

The following summarizes the signal model for threads:

A signal mask which specifies the signals blocked from delivery is associated with each thread.

The signal disposition, catch/ignore/default, is a process attribute and is shared by all threads in the process.

If the signal action for a signal specifies termination, stop or continue, all threads within the process are terminated, stopped or continued, respectively. This is the case regardless of whether the signal was directed at the process or a specific thread within the process.

Signals which are generated by some action associated with a particular thread, such as an invalid pointer dereference, will be delivered to the thread which caused the generation of the signal. These signals are referred to as synchronously generated signals.

Signals that are posted to the process by *kill(2)* or some asynchronous event such as terminal activity will be delivered to exactly one thread in the process which does not block delivery of the signal; if there is more than one eligible thread, which thread the signal is delivered to may not be able to be determined by an application. If all threads in the process block the signal, then the signal remains pending on the process until a thread unblocks the signal, issues a *sigwait()* call for the signal or sets the signal disposition to ignore the signal. These signals are referred to as asynchronously generated signals.

A thread can post a signal to a particular thread in the same process using *pthread_kill()*. If the thread which the signal is posted to blocks delivery of the signal, the signal remains pending on the thread.

The *sigpending()* function returns a union of the set of signals pending on the process and on the calling thread.

Each thread may define an alternate signal handling stack.

LWP Considerations

A signal mask which specifies the signals blocked from delivery is associated with each Lightweight Process (LWP).

The signal disposition, catch/ignore/default, is a process attribute and is shared by all LWPs in the process.

An LWP can post a signal to a particular LWP in the same process using *lwp_kill()*. If the thread which the signal is posted to blocks delivery of the signal, the signal remains pending on the thread.

Each LWP may define an alternate signal handling stack.

SEE ALSO

alarm(2), *ioctl(2)*, *kill(2)*, *sigaction(2)*, *sigaltstack(2)*, *siginterrupt(2)*, *signal(2)*, *sigpending(2)*, *sigproc-mask(2)*, *sigstack(2)*, *sigsuspend(2)*, *sigwait(2)*, *wait(2)*, *waitid(2)*.

CHANGE HISTORY

First released in Issue 1.

Issue 4

The following changes are incorporated for alignment with the ISO POSIX-1 standard:

- The function declarations in this header are expanded to full ISO C prototypes.
- The DESCRIPTION section is changed:
 - to define the type *sig_atomic_t*
 - to define the syntax of signal names and functions
 - to combine the two tables of constants
 - **SIGFPE** is no longer limited to floating-point exceptions, but covers all erroneous arithmetic operations.

The following change is incorporated for alignment with the ISO C standard:

- The **raise()** function is added to the list of functions declared in this header.

Other changes are incorporated as follows:

- A reference to **<sys/types.h>** is added for the definition of *pid_t*. This is marked as an extension.
- In the list of signals starting with **SIGCHLD**, the statement "but a system not supporting the job control option is not obliged to support the functionality of these signals" is removed. This is because job control is defined as mandatory on Issue 4 conforming implementations.
- Reference to implementation-dependent abnormal termination routines, such as creation of a core file, in item ii in the defaults action list is marked as an extension.

Issue 4, Version 2

The following changes are incorporated for X/OPEN UNIX conformance:

- The **SIGTRAP**, **SIGBUS**, **SIGSYS**, **SIGPOLL**, **SIGPROF**, **SIGXCPU**, **SIGXFSZ**, **SIGURG**, and **SIGVTALRM** signals are added to the list of signals that will be supported on all conforming

implementations.

- The *sa_sigaction* member is added to the **sigaction** structure, and a note is added that the storage used by *sa_handler* and *sa_sigaction* may overlap.
- The **SA_ONSTACK**, **SA_RESETHAND**, **SA_RESTART**, **SA_SIGINFO**, **SA_NOCLDWAIT**, **SS_ONSTACK**, **SS_DISABLE**, **MINSIGSTKSZ**, and **SIGSTKSZ** constants are defined. The **stack_t**, **sigstack**, and **siginfo_t** structures are defined.
- Definitions are given for the **ucontext_t**, **stack_t**, **sigstack**, and **siginfo_t** types.
- A table is provided listing macros that are defined as signal-specific reasons why a signal was generated. Signal-specific additional information is specified.
- The **bsd_signal()**, **killpg()**, **_longjmp()**, **_setjmp()**, **sigaltstack()**, **sig-hold()**, **sigrelse()**, **sigset()**, and **sigstack()** functions are added to the list of functions declared in this header.

HP-UX EXTENSIONS

DESCRIPTION

HP-UX supports multiple signal interfaces (see *sigaction(2)*, *signal(2)*, *sigvector(2)*, *bsdproc(3C)*, and *sigset(3C)*) that allow a process to specify the action taken upon receipt of a signal. All supported signal interfaces require specification of a signal, as designated by the **Name** and **Number** shown below. Signal specification can be any of the following except **SIGKILL** or **SIGSTOP**, which cannot be caught or ignored:

Name	Number	Notes	Meaning
SIGILL	04	A,B,C	illegal instruction
SIGTRAP	05	A,B,C	trace trap
SIGIOT	06	A,B	software generated signal
SIGEMT	07	A,B	software generated signal
SIGFPE	08	A,B	floating point exception
SIGKILL	09	A,D,E,F	kill
SIGCLD	18	G	death of a child (see WARNINGS below)
SIGPWR	19	C,G	power fail (see WARNINGS below)
SIGIO	22	G	asynchronous I/O signal; see <i>select(2)</i>
SIGWINCH	23	G	window size change; see <i>termio(7)</i>
SIGURG	29	G	urgent data arrived on an I/O channel
SIGLOST	30	A	file lock lost (NFS file locking)

The letters in the **Notes** column in the table above indicate the action taken when the signal is received, and any special conditions on its use:

- A* The default action is to terminate the process.
- B* The default action of terminating the process also generates a core image file if possible.
- C* The action is not reset to SIG_DFL before calling the signal-catching function.
- D* The signal cannot be ignored.
- E* The signal cannot be caught.
- F* The signal will not be held off from a stopped process.
- G* The default action is to ignore the signal.
- H* The default action is to stop the process.

All signal interfaces allow specification of an *action* that determines what to do upon the receipt of a signal, and should be one of the following:

SIG_DFL Execute the default action, which varies depending on the signal as described above:

- A* Terminate the receiving process with all of the consequences outlined in *exit(2)*.
- B* If following conditions are met, generate a core image file (see *core(4)*) in the current working directory of the receiving process:
 - The effective user ID and the real user ID of the receiving process are equal.
 - The effective group ID and the real group ID of the receiving process are equal.
 - A regular file named **core** does not exist and can be created, or exists and is writable.

If the file is created, it has the following properties:

- The file mode is 0666, modified by the file creation mode mask (see *umask(2)*).
- The file user ID is equal to the effective user ID of the receiving process.
- The file group ID is equal to the effective group ID of the receiving process.

- G* Ignore the signal. Do not terminate or stop the receiving process.
- H* Stop the receiving process. While a process is stopped, any additional signals sent to the process are suspended until the process is restarted (except those marked with Note *F* above, which are processed immediately). However, when the process is restarted, pending signals are processed. When a process that is in an orphaned process group (see *glossary*(9)) receives a **SIGTSTP**, **SIGTTIN**, or **SIGTTOU** signal, the process is not stopped because a process in an orphaned process group is not allowed to stop. Instead, a **SIGHUP** signal is sent to the process, and the **SIGTSTP**, **SIGTTIN**, or **SIGTTOU** is discarded.

SIG_IGN Ignore the signal.

When one of the supported signal interface routines is used to set the action of a signal to **SIG_IGN** and an instance of the signal is pending, the pending signal is cleared.

- D* Signals marked with Note *D* above cannot be ignored.

address Catch the signal.

Upon receipt of the signal, if **signal()** is used to set the action, reset the action for the signal caught to **SIG_DFL** (except signals marked with Note *C*). Then, call the signal-catching function to which *address* points, and resume executing the receiving process at the point where it was interrupted. Signal interface routines other than **signal()** normally do not reset the action for the signal caught. However, **sigaction()** and **sigvector()** provide a way of specifying this behavior (see *sigaction*(2) or *sigvector*(2)).

The signal-catching function is called with the following three parameters:

- sig* The signal number.
- code* A word of information usually provided by the hardware.
- scp* A pointer to the machine-dependent structure *sigcontext* defined in **<signal.h>**.

Depending on the value of *sig*, *code* can be zero and/or *scp* can be NULL. The meanings of *code* and *scp* and the conditions determining when they are other than zero or NULL are implementation-dependent (see **DEPENDENCIES** below). It is possible for *code* to always be zero, and *scp* to always be NULL.

The pointer *scp* is valid only during the context of the signal-catching function.

Optional parameters can be omitted from the signal-catching function parameter list, in which case the signal-catching function is exactly compatible with UNIX System V. Truly portable software should not use the optional parameters in signal-catching routines.

Upon return from the signal-catching function, the receiving process resumes execution at the point where it was interrupted.

When a signal is caught during the execution of system calls such as **read()**, **write()**, **open()**, or **ioctl()** on a slow device (such as a terminal, but not a file), during a **pause()** system call or a **wait()** system call that does not return immediately because a previously stopped or zombie process already exists, the signal-catching function is executed and the interrupted system call returns a -1 to the calling process with **errno** set to **EINTR**.

- C* If the signal is marked with Note *C* above, the action is not reset to **SIG_DFL** before calling the signal-catching function. Furthermore, the action is not reset if any signal interface routine other than **signal()** was used to set the action. See the description of signal catching above.

- E* If the signal is marked with Note *E* above, the signal cannot be caught.

When any stop signal (**SIGSTOP**, **SIGTSTP**, **SIGTTIN**, **SIGTTOU**) is generated for a process, pending **SIGCONT** signals for that process are discarded. Conversely, when **SIGCONT** is generated for a process, all pending stop signals for that process are discarded. When **SIGCONT** is generated for a stopped process, the process is continued, even if the **SIGCONT** signal is blocked or ignored. If **SIGCONT** is blocked and not ignored, the process remains pending until it is either unblocked or a stop signal is generated.

SIGKILL is sent by the system if an **exec()** system call is unsuccessful and the original program has already been deleted.

WARNINGS

The signals **SIGCLD** and **SIGPWR** behave differently than those described above.

The actions for these signals is modified as follows:

SIGCLD Setting the action for **SIGCLD** to **SIG_IGN** in a parent process prevents exiting children of the calling process from creating a zombie process. If the parent process executes the **wait()** function, the calling process blocks until all of the child processes of the calling processes terminate. The **wait()** function then returns a value of **-1** with **errno** set to **ECHILD** (see *wait(2)*).

If one of the signal interface routines is used to set the action for **SIGCLD** to be caught (that is, a function address is supplied) in a process that currently has terminated (zombie) children, a **SIGCLD** signal is delivered to the parent process immediately. Thus, if the signal-catching function reinstalls itself, the apparent effect is that any **SIGCLD** signals received due to the death of children while the function is executing are queued and the signal-catching function is continually reentered until the queue is empty. Note that the function must reinstall itself after it calls **wait()**, **wait3()**, or **waitpid()**. Otherwise the presence of the child that caused the original signal causes another signal immediately, resulting in infinite recursion.

When processing a pipeline, the Bourne shell (see *sh-bourne(1)*) makes the last process in the pipeline the parent of the preceding processes. Job control shells including C shell, Korn shell and the POSIX shell (see *csh(1)*, *ksh(1)*, and *sh-posix(1)*) make the shell itself the parent of all processes in the pipeline. Therefore, a process that can receive data from a pipe should not attempt to catch **SIGCLD**.

SIGPWR The **SIGPWR** signal is sent to all processes after a power interruption when power is restored and the system has done all necessary reinitialization. Processes restart by catching (or ignoring) **SIGPWR**.

Applications that wish to recover from power failures should catch **SIGPWR** and take whatever necessary steps to reinitialize itself.

Some implementations do not generate **SIGPWR**. Only systems with nonvolatile memory can recover from power failures.

DEPENDENCIES**Series 700**

The signal **SIGPWR** is not currently generated.

Series 700/800

The structure pointer *scp* is always defined.

The *code* word is always zero for all signals except **SIGILL** and **SIGFPE**. For **SIGILL**, *code* has the following values:

- 8 Illegal instruction trap;
- 9 Break instruction trap;
- 10 Privileged operation trap;
- 11 Privileged register trap.

For **SIGFPE**, *code* has the following values:

- 12 Overflow trap;
- 13 Conditional trap;
- 14 Assist exception trap;
- 22 Assist emulation trap.

Refer to the Series 800 processor documentation provided with your system for more detailed information about the meaning of these errors.

The Instruction Address Offset Queue (program counter) is not advanced when a trap occurs on Series 800 systems. If a signal generated by a hardware trap is masked or has its signal action set to **SIG_IGN**, the program loops infinitely since the instruction causing the trap is re-executed, causing the trap again. If the signal is received by a signal-catching function in the user program, the instruction that caused the trap is

re-executed upon return from the signal-catching function unless program flow is altered by the signal-catching function. For example, the **longjmp()** routine (see *setjmp(3C)*) can be called. Using **longjmp()** ensures software portability across different hardware architectures.

AUTHOR

signal was developed by HP, AT&T, and the University of California, Berkeley.

SEE ALSO

kill(1), init(1M), bsdproc(3C), exit(2), kill(2), lseek(2), pause(2), sigaction(2), signal(2), sigvector(2), wait(2), sigset(3C), abort(3C), setjmp(3C).

STANDARDS CONFORMANCE

<**signal.h**>: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1, ANSI C

NAME

sis - secure internet services with Kerberos authentication and authorization

DESCRIPTION

Secure Internet Services (SIS) provides network authentication when used in conjunction with HP DCE security services, the HP Praesidium/Security Server, or other software products that provide a Kerberos V5 Network Authentication Services environment. The network authentication ensures that a local and remote host will be mutually identified to each other in a secure and trusted manner and that the user is authorized to use the service on the remote host.

Traditional internet services such as **telnet**, **rlogin**, or **ftp**, allow the user to access remote systems by typing a password that is then transmitted to the remote system over the network. The password is transmitted without encryption over the network, permitting an observer to capture the cleartext packets containing the password. This has been a major security hole for traditional internet services.

The optional Secure Internet Services are a replacement for their traditional counterparts and prevent the cleartext transmission of user passwords over the network. However, none of these services will encrypt the session beyond what is necessary to authenticate the service or authorize the user.

This man page assumes the reader is familiar with Kerberos terminology normally provided with your Kerberos V5 Network Authentication Services environment. The intent here is to describe those aspects of the Kerberos environment specifically used by SIS.

Authentication

For Kerberos authentication to succeed, the user must have successfully logged into a system within the Kerberos realm and obtained a set of credentials. The credentials include a Ticket Granting Ticket (TGT) and a session key. The SIS client will use the TGT to obtain a service ticket to access a SIS daemon on the network. If the credentials are missing or the TGT is invalid, the authentication will fail and connection to the SIS daemon will be denied.

For systems configured into a DCE cell, credentials are obtained through the **dce_login** command. For systems configured into a Praesidium/Security Server cell, credentials are obtained through the **dess_login** command. In a non-DCE Kerberos-based secure environment, credentials are obtained through the **kinit** command.

Authorization

For every user of these services, a user principal must be configured into the Key Distribution Center's database. The user principal allows the user to obtain a service ticket which is sent to the remote service as part of the Kerberos authentication mechanism. If the authentication is successful, the user principal is then used as part of the Kerberos authorization mechanism.

In order for the authorization to succeed, both of the following requirements must be met:

1. The login name must exist in the remote system's password file, i.e., the remote account must exist. Note: the login name is the name specified by the user in response to a login prompt and may be different from the current user name.
2. One of the following conditions must be true:
 - A. The remote account's home directory has a **.k5login** file that contains the user principal. The **.k5login** file must be owned by that account and only that account can have write permission (i.e., the permissions would appear as **-rw-r--r--**).
 - B. The remote system has an authorization name database file, **aname**, that contains the user principal. The **aname** file should contain a mapping of the user principal to an account on the remote system.
 - C. The user name in the user principal is the same as the user name of the account being accessed, and the local and remote systems are in the same realm.

If authorization succeeds, the user will not see a prompt for a password (when a password is required) and the connection to the remote system will succeed. If the authentication or authorization fails, the user will be notified of the error and will not be allowed to continue.

Bypassing or Enforcing Authentication/Authorization

If the authentication or authorization fails, the service can be re-run with a special command line option (**-P**) to request non-Kerberos authentication. However, when a password is required, it will be sent across the network in a readable form. Typically, this special command line option should only be used to access

non-secure remote systems.

The **ftp** and **telnet** daemons have a special command line option (**-A**) which can be used to ensure that non-secure systems are denied access.

To prevent non-secure access through the **rcp**, **remsh** or **rlogin** commands, the **inetd.conf** file on the remote system should be edited to comment out the entries for **shell** and **login**.

SERVICES

ftp , ftpd	file transfer program
rlogin , rlogind	remote login
telnet , telnetd	user interface and server for Telnet protocol
rcp , remshd	remote file copy
remsh , remshd	execute from a remote shell

TROUBLESHOOTING

For the correct execution of SIS, it is important that the secure environment be properly installed, configured and running. The following is a quick checklist to verify this:

1. The DCE, Praesidium/Security Server, or Kerberos security system should be running on the Kerberos server. The **/etc/services** file should contain entries for the Kerberos ports.
2. The user's user principal must be entered into the Key Distribution Center's database. Use the appropriate tool (e.g., **kadmin** or HP DCE's **dcecp**) to list the database and to verify that the user has a user principal configured.
3. The Kerberos configuration directory on the local and remote systems should contain a **krb.conf**, **krb.realms**, and a server key table file. Generally, the Kerberos configuration directory will be **/krb5** and the server key table file will be named **v5srvtab**.
4. The user principal must be specified in **~/k5login** on the local and remote systems. The **~/k5login** lists the principals and realm names which have access permission for the user's account.

Alternatively, the secure system can use an authorization name database, **aname**, on the local and remote systems. An entry in this file will authorize the user name in a user principal to the specified login name.

Verify that **~/k5login** exists, has the correct permissions (i.e., **-rw-r--**), and includes the user principal. Or, use the appropriate tool (e.g., **krb5_anadd** on a non-HP DCE system) to verify that the user principal is included in the **aname** file.

5. The server key table file on the remote system should contain a host principal. The root user can verify the contents of the **v5srvtab** through the command: **klist -k**. If **klist** supports the **-k** option, type this command and verify that a host principal is listed.

Alternatively, if the validation tool, **krbval**, is available on the system, use the command: **krbval -v**.

6. If **krbval** is available on the local and remote systems, use it to test the Kerberos configuration by invoking it to act as a client application on the local system and a server application on the remote system. See **krbval(1M)** for details.
7. The SIS files must be installed. The traditional services will have been saved and the files for the new services will be linked to the original, traditional file names.

DIAGNOSTICS

In addition to Kerberos-specific error messages, SIS has a few security related error messages that are common to several or all of the services. These error messages can be used by scripts to detect whether the invocation of a service has failed.

Error and warning messages reported by the SIS clients

ERROR! Kerberos authentication failed.

The user has not obtained a valid Ticket Granting Ticket (through **kinit**, **dce_login**, or **dess_login**) or a valid host principal has not been configured in the Key Distribution Center's database for the realm. A more specific error message indicating the possible cause of the failure will

accompany this error message.

This error message will also be generated if the user attempts to access a non-secure remote system. In which case, this message will be preceded by the message: "To bypass Kerberos authentication, use the -P option".

This error is reported by ftp, rlogin and telnet.

ERROR! Kerberos-specific options are invalid with the -P option.

The -P command line option indicates that Kerberos authentication should not be performed. If any Kerberos-specific options are also specified on the command line, then they are in contradiction to this request.

For **remsh** and **rlogin**, this means the -P option can not be used in conjunction with the -F, -f or -k options.

For **rcp** this means the -P option can not be used in conjunction with the -k option.

For **telnet**, this means the -P option cannot be used in conjunction with the -a or -l options.

WARNING! Password will be sent in a non-secure manner.

WARNING! Kerberos authentication will be bypassed.

The user has specified the -P option on the command line to access a non-secure remote system or to bypass a bad configuration in the Kerberos environment.

In the cases where a password is requested, the -P command line option will cause the password to be sent across the network in a readable form where it could possibly be intercepted or captured.

It is recommended that the user corrects a bad configuration and only uses the -P option if the remote system is non-secure.

The first warning is reported by **ftp**, **rlogin**, and **telnet**. The second warning is reported by **rcp**. **remsh** could report either warning depending upon whether a password is required.

Error messages reported in the syslog by the SIS daemons

ERROR! Access denied. Kerberos authentication must succeed.

The daemon was started with the -A command line switch to ensure that non-secure access by remote systems will be denied. The user cannot access the remote system unless the local system has been configured for secure access.

This error is logged by **ftpd** and **telnetd**.

ERROR! Principal <principal> (<remote_user>@<remote_host>) logging in as <local_user> has no account.

The "local_user" does not have a valid password file entry.

This error is logged by all SIS daemons.

ERROR! Principal <principal> (<remote_user>@<remote_host>) logging in as <local_user> failed krb5_userok.

Authentication succeeded but authorization failed. The user should verify that their user name is listed in `~/k5login` or in the `aname` file on the remote system. The user's `~/k5login` must have the correct permissions and must be owned by the user (i.e., `-rw-r--r--`).

This error is logged by all SIS daemons.

ERROR! Principal <principal> (<remote_user>@<remote_host>) logging in as <local_user> failed ruserok.

The `/etc/hosts.equiv` or `~/rhost` files are missing or are not set up properly to authorize "local_user" (see `ruserok(3N)`).

This error is logged by `rlogind` or `remshd` if they are started with the -R, -r or -k options.

SEE ALSO

`ftp(1)`, `kinit(1)`, `kdestroy(1)`, `klist(1)`, `krbval(1M)`, `rcp(1)`, `remsh(1)`, `rlogin(1)`, `telnet(1)`, `dce_intro(1M)`, `dce_login(1M)`, `dess_login(1M)`, `ftpd(1M)`, `remshd(1M)`, `rlogind(1M)`, `telnetd(1M)`, `dess(5)`.

NAME

sys/stat.h - data returned by the stat() function

SYNOPSIS

```
#include <sys/stat.h>
```

DESCRIPTION

The `<sys/stat.h>` header defines the structure of the data returned by the functions `fstat()`, `lstat()`, and `stat()`. The structure `stat` contains at least the following members:

<code>dev_t</code>	<code>st_dev</code>	ID of device containing file
<code>ino_t</code>	<code>st_ino</code>	file serial number
<code>mode_t</code>	<code>st_mode</code>	mode of file (see below)
<code>nlink_t</code>	<code>st_nlink</code>	number of links to the file
<code>uid_t</code>	<code>st_uid</code>	user ID of file
<code>gid_t</code>	<code>st_gid</code>	group ID of file
<code>dev_t</code>	<code>st_rdev</code>	device ID (if file is character or block special)
<code>off_t</code>	<code>st_size</code>	file size in bytes (if file is a regular file)
<code>time_t</code>	<code>st_atime</code>	time of last access
<code>time_t</code>	<code>st_mtime</code>	time of last data modification
<code>time_t</code>	<code>st_ctime</code>	time of last status change
<code>long</code>	<code>st_blksize</code>	a filesystem-specific preferred I/O block size for this object. In some filesystem types, this may vary from file to file
<code>long</code>	<code>st_blocks</code>	number of blocks of a filesystem-specific size allocated for this object

File serial number and device ID taken together uniquely identify the file within the system. The `dev_t`, `ino_t`, `mode_t`, `nlink_t`, `uid_t`, `gid_t`, `off_t`, and `time_t` types are defined as described in `<sys/types.h>`. Times are given in seconds since the Epoch.

The following symbolic names for the values of `st_mode` are also defined:

File type:

<code>S_IFMT</code>	type of file
<code>S_IFBLK</code>	block special
<code>S_IFCHR</code>	character special
<code>S_IFIFO</code>	FIFO special
<code>S_IFREG</code>	regular
<code>S_IFDIR</code>	directory
<code>S_IFLNK</code>	symbolic link

File mode bits:

<code>S_IRWXU</code>	read, write, execute/search by owner
<code>S_IRUSR</code>	read permission, owner
<code>S_IWUSR</code>	write permission, owner
<code>S_IXUSR</code>	execute/search permission, owner
<code>S_IRWXG</code>	read, write, execute/search by group
<code>S_IRGRP</code>	read permission, group
<code>S_IWGRP</code>	write permission, group
<code>S_IXGRP</code>	execute/search permission, group
<code>S_IRWXO</code>	read, write, execute/search by others
<code>S_IROTH</code>	read permission, others
<code>S_IWOTH</code>	write permission, others
<code>S_IXOTH</code>	execute/search permission, others
<code>S_ISUID</code>	set-user-ID on execution
<code>S_ISGID</code>	set-group-ID on execution
<code>S_ISVTX</code>	on directories, restricted deletion flag

The bits defined by `S_IRUSR`, `S_IWUSR`, `S_IXUSR`, `S_IRGRP`, `S_IWGRP`, `S_IXGRP`, `S_IROTH`, `S_IWOTH`, `S_IXOTH`, `S_ISUID`, `S_ISGID` and `S_ISVTX` are unique. `S_IRWXU` is the bitwise OR of `S_IRUSR`, `S_IWUSR`, and `S_IXUSR`. `S_IRWXG` is the bitwise OR of `S_IRGRP`, `S_IWGRP`, and `S_IXGRP`. `S_IRWXO` is the bitwise OR of `S_IROTH`, `S_IWOTH`, and `S_IXOTH`.

Implementations may OR other implementation-dependent bits into `S_IRWXU`, `S_IRWXG`, and `S_IRWXO`, but they will not overlap any of the other bits defined in this document. The file permission bits are defined to be those corresponding to the bitwise inclusive OR of `S_IRWXU`, `S_IRWXG`, and `S_IRWXO`.

The following macros will test whether a file is of the specified type. The value *m* supplied to the macros is the value of *st_mode* from a `stat` structure. The macro evaluates to a non-zero value if the test is true, 0 if the test is false.

```
S_ISBLK(m)    Test for a block special file.
S_ISCHR(m)    Test for a character special file.
S_ISDIR(m)    Test for a directory.

S_ISFIFO(m)   Test for a pipe or FIFO special file.
S_ISREG(m)    Test for a regular file.
S_ISLNK(m)    Test for a symbolic link.
```

The following are declared as functions and may also be defined as macros:

```
int    chmod(const char *path, mode_t mode);
int    lstat(const char *path, struct stat *buf);
int    mkdir(const char *path, mode_t mode);
int    mkfifo(const char *path, mode_t mode);
int    mknod(const char *path, mode_t mode, dev_t dev);
int    stat(const char *path, struct stat *buf);
mode_t umask(mode_t cmask);
```

APPLICATION USAGE

Use of the macros is recommended for determining the type of a file.

SEE ALSO

`chmod()`, `fchmod()`, `fstat()`, `lstat()`, `mkdir()`, `mkfifo()`, `mknod()`, `stat()`, `umask()`, `<sys/types.h>`.

CHANGE HISTORY

First released in Issue 1.

Derived from Issue 1 of the SVID.

Issue 4

The following changes are incorporated for alignment with the ISO POSIX-1 standard:

- The function declarations in this header are expanded to full ISO C prototypes.
- The DESCRIPTION section is expanded to indicate (a) how files are uniquely identified within the system, (b) that times are given in units of seconds since the Epoch, (c) rules governing the definition and use of the file mode bits, and (d) usage of the file type test macros.

Other changes are incorporated as follows:

- Reference to the header `<sys/types.h>` is added for the definitions of *dev_t*, *ino_t*, *mode_t*, *nlink_t*, *uid_t*, *gid_t*, *off_t*, and *time_t*. This has been marked as an extension.
- References to the `S_IREAD`, `S_IWRITE`, `S_IEXEC` file and `S_ISVTX` modes are removed.
- The descriptions of the members of the `stat` structure in the DESCRIPTION section are corrected.

Issue 4, Version 2

The following changes are incorporated for X/OPEN UNIX conformance:

- The *st_blksize* and *st_blocks* members are added to the `stat` structure.
- The `S_IFLINK` value of `S_IFMT` is defined.
- The `S_ISVTX` file mode bit and the `S_ISLNK` file type test macro is defined.
- The `fchmod()`, `lstat()`, and `mknod()` functions are added to the list of functions declared in this header.

HP-UX EXTENSIONS

NAME

stat.h - file-specific information

DESCRIPTIONThe contents of the *stat* structure include the following members:

```

    ushort st_fstype;    /*Type of filesystem this file */
                        /* is in; see vfstmount(2) */

    dev_t   st_realdev;   /* Real device number of device */
                        /* containing the inode for this file */

```

The following symbolic names for the values of the *st_mode* field are defined as indicated:

File type:

```

S_IFMT    0170000 type of file
S_IFSOCK  0140000 socket
S_IFLNK   0120000 symbolic link
S_IFNWK   0110000 network special
S_IFREG   0100000 regular (ordinary)
S_IFBLK   0060000 block special
S_IFDIR   0040000 directory
S_IFCHR   0020000 character special
S_IFIFO   0010000 FIFO special (named pipe)

```

File mode bits:

File miscellaneous mode bits:

```

S_CDF     0004000 directory is a context-dependent file
S_ISUID   0004000 set user id on execution
S_ISGID   0002000 set group id on execution
S_ENFMT   0002000 set file-locking mode to enforced
S_ISVTX   0001000 save swapped text even after use

```

File permission mode bits:

```

S_IRWXU   0000700 owner's file access permission bits
S_IRUSR   0000400 read access permission for owner
S_IWUSR   0000200 write access permission for owner
S_IXUSR   0000100 execute/search access permission for owner
S_IRWXG   0000070 group's file access permission bits
S_IRGRP   0000040 read access permission for group
S_IWGRP   0000020 write access permission for group
S_IXGRP   0000010 execute/search access permission for group
S_IRWXO   0000007 others' access permission bits
S_IROTH   0000004 read access permission for others
S_IWOTH   0000002 write access permission for others
S_IXOTH   0000001 execute/search access permission for others

```

Obsolete names for file permission mode bits:

```

S_IREAD   0000400 read access permission for owner
S_IWRITE  0000200 write access permission for owner
S_IEXEC   0000100 execute/search access permission for owner

```

File type test macros:

```

S_ISCDF(m) test for a context-dependent file

```

S_ISNWK(*m*) test for a network special
S_ISSOCK(*m*) test for a socket

SEE ALSO

chmod(2), chown(2), link(2), mkdir(2), mkfifo(2), mknod(2), stat(2), symlink(2), umask(2), utime(2), types(5).

STANDARDS CONFORMANCE

<sys/stat.h>: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

stdarg.h - macros for handling variable argument lists

SYNOPSIS

```
#include <stdarg.h>

void va_start(va_list pvar, argN);
type va_arg(va_list pvar, type);
void va_end(va_list pvar);
```

DESCRIPTION

The **<stdarg.h>** header contains a set of macros that can be used to write portable procedures that accept variable argument lists. Routines that have variable argument lists (such as **printf()**) but do not use *stdarg* are inherently nonportable, because different machines use different argument-passing conventions.

va_list is a type defined for the variable used to traverse the list.

va_start is called to initialize *pvar* to the beginning of the list. The type of *argN* should be the same as the argument to the function just before the variable portion of the argument list.

va_arg returns the next argument in the list pointed to by *pvar*. *type* is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, because it cannot be determined at runtime.

va_end is used to clean up.

Multiple traversals, each bracketed by *va_start* ... *va_end*, are possible.

EXAMPLE

This example is a possible implementation of **exec1** (see *exec(2)*):

```
#include <stdarg.h>
#define MAXARGS 100

/* exec1 is called by
   exec1(file, arg1, arg2, ..., (char *)0);
*/
exec1(const char *file, const char *args, ...)
{
    va_list ap;
    char *array[MAXARGS];
    int argno = 0;

    va_start(ap, args);
    if ((array[0] = args) != 0)
        while ((array[argno++] = va_arg(ap, char *)) != 0)
            ;
    va_end(ap);
    return execv(file, array);
}
```

SEE ALSO

exec(2), *vprintf(3S)*, *varargs(5)*.

WARNINGS

It is up to the calling routine to specify how many arguments there are, since it is not always possible to determine this from the stack frame. For example, **exec1()** is passed a zero pointer to signal the end of the list, and **printf()** can tell how many arguments are there by the format string.

Unless ANSI C is used, it is non-portable to specify a second argument of *char*, *short*, or *float* to *va_arg*, because arguments seen by the called function are never *char*, *short*, or *float*.

Pre-ANSI C converts *char* and *short* arguments to *int* and converts *float* arguments to *double* before passing them to a function.

STANDARDS CONFORMANCE

<stdarg.h>: AES, SVID3, XPG4, FIPS 151-2, POSIX.1, ANSI C

va_arg: SVID3, XPG4, ANSI C

va_end: SVID3, XPG4, ANSI C

va_list: SVID3, XPG4, ANSI C

va_start: SVID3, XPG4, ANSI C

NAME

stdsyms - description of HP-UX header file organization

DESCRIPTION

HP-UX header files are organized in a manner that allows for only a subset of the symbols available in that header file to be visible to an application that conforms to a specific standard. The ANSI-C, POSIX.1, POSIX.2, and XPG4 standards each reserve a certain set of symbols for that standard's namespace. In addition, the HP-UX implementation of XPG3 and the "OSF AES/OS" provides for a clean namespace although this is not a specific requirement of those standards.

The following rules apply in determining what symbols are reserved for any standard. These symbols are reserved for the standard and for use by the implementation, and must be either avoided altogether, or used exactly as defined by the specified standard.

- All symbols defined by the desired standard are reserved. Refer to the appropriate standards documentation for a complete list of reserved symbols.
- All symbols beginning with an underscore followed by another underscore or an uppercase letter are reserved for the implementation.
- All external identifiers beginning with an underscore are reserved for the implementation.

The following is a list of feature test macros which must be defined to obtain the appropriate namespace from the header files.

_STD_C_

This symbol is automatically defined by the ANSI-C pre-processor (`/opt/langtools/lib/cpp.ansi`) and is automatically defined when specifying an ANSI-C compile (`cc -Aa`). Using the strict ANSI option `-Aa` requests a pure ANSI-C namespace, which is the smallest subset of the HP-UX namespace available. The `-Aa` option also enables the inclusion of ANSI-C-style function prototypes for increased type checking. Note that the default namespace when using the `-Aa` option is the ANSI-C namespace; therefore a broader namespace must be *requested* if it is desired.

_POSIX_SOURCE

As documented in the IEEE POSIX.1 standard, the programmer is required to define the `_POSIX_SOURCE` feature test macro to obtain the POSIX.1 namespace and POSIX.1 functionality. This feature test macro can be defined, either by using compiler options (`-D_POSIX_SOURCE`) or by using `#define` directives in the source files before any `#include` directives. Note that the default POSIX namespace is the POSIX.1-1990 namespace. It is necessary to define the `_POSIX1_1988` feature test macro in addition to the `_POSIX_SOURCE` macro in order to obtain the POSIX.1-1988 namespace.

_POSIX_C_SOURCE

As documented in the IEEE POSIX.2 standard, the programmer is required to define the `_POSIX_C_SOURCE` feature test macro with a value of 2 to obtain the POSIX.1 and POSIX.2 namespaces and functionality. This feature test macro can be defined, either by using compiler options (`-D_POSIX_C_SOURCE=2`) or by using `#define` directives in the source files before any `#include` directives. This macro is also automatically defined in the XPG4 X/Open namespace (that is, whenever `_XOPEN_SOURCE` and `_XPG4` are defined without defining `_XPG2` or `_XPG3`).

_XOPEN_SOURCE

As documented in the XPG3 and XPG4 standards, the programmer is required to define the `_XOPEN_SOURCE` feature test macro to obtain X/Open functionality. This feature test macro can be defined, either by using compiler options (`-D_XOPEN_SOURCE`) or by using `#define` directives in the source files before any `#include` directives. Although XPG3 does not specify any namespace pollution rules, XPG4 has instituted such rules. Therefore, the HP-UX operating system provides clean namespaces whenever `_XOPEN_SOURCE` is defined.

The current default X/Open namespace is that corresponding to XPG3. To request other versions of the X/Open namespace, define `_XPG2` or `_XPG4` in conjunction with `_XOPEN_SOURCE`.

_AES_SOURCE

As documented in the "OSF AES/OS" standard, the programmer is required to define the `_AES_SOURCE` feature test macro to obtain OSF functionality. This feature test macro

can be defined, either by using compiler options (`-D_AES_SOURCE`) or by using `#define` directives in the source files before any `#include` directives. Although the AES does not specify any namespace pollution rules, the other standards have instituted such rules. Therefore HP-UX provides a clean namespace whenever `_AES_SOURCE` is defined.

`_HPUX_SOURCE`

The programmer can define the `_HPUX_SOURCE` feature test macro to obtain the HP-UX namespace and complete HP-UX functionality. Note that the HP-UX namespace is currently a superset of all of the above mentioned namespaces. When using the compatibility-mode compiler (`cc(1)` without the `-Aa` option), the HP-UX namespace is provided by default. The programmer must request one of the other namespaces as described above to obtain the appropriate subset of the HP-UX namespace. When using the strict ANSI-C-mode compiler (`cc -Aa`), the programmer must specifically request a broader namespace.

The `_HPUX_SOURCE` feature test macro can be defined, either by using compiler options (`-D_HPUX_SOURCE`) or by using `#define` directives in the source files before any `#include` directives.

The following is a list of miscellaneous feature test macros that provide various additional features.

`__cplusplus`

This symbol is automatically defined by the HP C++ compiler. Defining this macro enables the C++ function prototypes in system header files.

The default namespace for HP C++ is the ANSI-C namespace. To obtain another namespace define the appropriate feature test macro.

HP C++ uses the ANSI-C preprocessor by default. To get the compatibility mode preprocessor, use the `-Ac` option to `cc(1)`. The compatibility mode preprocessor uses the HP-UX namespace (`_HPUX_SOURCE`).

`_POSIX1_1988`

This feature test macro should be defined when the `POSIX.1-1988` namespace is required. It should be used in conjunction with the `_POSIX_SOURCE` macro if the default `POSIX.1-1990` namespace is not desired.

This macro is defined automatically whenever `_AES_SOURCE` or `_XPG3` is requested.

`_XPG2`

The `_XPG2` macro can be defined when using the compatibility-mode compiler to obtain XPG2 functionality. This provides XPG2 specified function declarations and macros in the HP-UX namespace. Note that the values obtained from most of the macros available when using this option are now available at run-time via the `pathconf()`, `fpathconf()`, and `sysconf()` system calls (see `pathconf(2)` and `sysconf(2)`). Use of the `_XPG2` macro is strongly discouraged because it gives access to obsolete functionality. Note that no function prototypes are provided when using this feature test macro.

`_XPG3`

The `_XPG3` feature test macro is defined automatically if the programmer has requested the XPG3 namespace (i.e., defined `_XOPEN_SOURCE`, but not some other conflicting namespace such as `_XPG2` or `_XPG4`).

`_XPG4`

The `_XPG4` feature test macro is provided so that the programmer can obtain the XPG4 namespace, since it differs slightly from the `_XPG3` namespace. In order to obtain the XPG4 namespace, the programmer must define both the `_XOPEN_SOURCE` and `_XPG4` feature test macros. The `_XOPEN_SOURCE` and `_XPG4` feature test macros can be defined, either by using compiler options (`-D_XOPEN_SOURCE` `-D_XPG4`) or by using `#define` directives in the source files before any `#include` directives.

`_SVID2`

The `_SVID2` macro can be defined when using the compatibility mode compiler to obtain SVID2 function return types in the HP-UX namespace. The default return types of many functions have since been changed in the HP-UX operating system to align with the ANSI-C, POSIX, X/Open, and OSF standards.

`_CLASSIC_TYPES`

The `_CLASSIC_TYPES` macro can be defined by the programmer to obtain pre-7.0 style function return types and structure element types. This macro has been provided only as a transition aid when migrating from the pre-7.0 version of HP-UX to standards-based HP-UX. Use of this macro is strongly discouraged as this functionality will be removed in a

future release of HP-UX. Note that no function prototypes are provided when using this feature test macro.

SEE ALSO

cc(1), cpp(1), pathconf(2), sysconf(2).

NAME

suffix - file-name suffix conventions

DESCRIPTION

The following list summarizes file name suffix conventions that can be found in an HP-UX system. It is a partial compilation of possibly useful knowledge, suggestions, and explanations, rather than a specification of standards. Suffixes are often used in preference to prefixes because they enable related files to group together alphabetically in a directory listing.

Note that some programs require the use of a specific value, or vary their behavior based on a choice of suffixes. Such programs are noted in many (but not all) cases.

.A	HP64000 cross assembler symbol file.
.a	Library file (archive) managed by <i>ar</i> ; known to <i>make</i> .
.ad?	HP Ada source, where "?" stands for any single character.
.allow	List of users allowed by <i>at</i> or <i>cron</i> (for example, at.allow).
.an	Source for nroff "man" macros.
.ASC	LIF (Logical Interchange Format) type 1, ASCII file for use by Pascal or BASIX/UX. Incompatible with <i>lifcp</i> .
.aux	Cross-referencing information created automatically by LaTeX.
.awk	<i>awk</i> script file.
.b	Compiled LISP (.l) source file, or a bold font file.
.back	
.bak	
.bkup	Backup copy of a file.
.BAD	
.bad	File containing bad data, or occupying a bad spot on a disk.
.bbl	Bibliography created by BibTeX for inclusion in a LaTeX document.
.bib	Bibliographic data file, (for example, BibTeX bibliography database).
.blg	Log of errors from BibTeX.
.bst	BibTeX bibliography style definition.
.C	File compressed by <i>compact</i> , or C++ language source file, or HP64000 cross compiled C source file.
.c	C language source file; known to <i>cc</i> and <i>make</i> .
.cas	CAST language scripts.
.cat	NLS (Native Language Support) message catalog.
.cf	Configuration file (for example, sendmail.cf).
.clu	CLU file.
.CODE	Pascal workstation object code.
.cpio	File containing output from <i>cpio -o</i> , that is, a <i>cpio</i> archive.
.csh	C-shell (<i>csh</i>) script.
.curr	Current version of a file.
.d	Directory file, or data file.
.day	A script that is read daily.
.deny	List of users denied by <i>at</i> or <i>cron</i> (for example, cron.deny).
.devs	List of devices.
.diff	Differences between two files, output from <i>diff</i> .

.dir	DBM database directory file.
.doc	Documentation file of some sort.
.dvi	Device-independent text formatter output.
.e	Extended FORTRAN language (EFL) source file; known to <i>make</i> .
.el	GNU Emacs Lisp file.
.elc	Compiled GNU Emacs Lisp file.
.eqn	Source for <i>nroff</i> equation macros.
.err	Standard error from a program.
.errors	
.errs	Errors recorded by a program.
.f	FORTRAN language source file; known to <i>fc</i> and <i>make</i> .
.f77	FORTRAN 77 language source file.
.fc	Frozen configuration file (for example, sendmail.fc).
.full	A complete file or list.
.gf	TeX font bitmaps in Generic Font format.
.glo	Glossary created by LaTeX.
.h	C language header (include) file; known to <i>make</i> .
.help	
.hf	
.hlp	Help text for a program, often read automatically.
.hour	
.hr	A script that is read hourly.
.i	Output of C preprocessor ("cc -P"), or a Berkeley Pascal language include file, or an italicized font file.
.icn	Icon source code.
.idx	Index created by LaTeX.
.in	Standard input to a program.
.INDEX	<i>notes</i> index file.
.ksh	Korn shell script file.
.L	HP64000 cross linker symbol file.
.l	<i>lex</i> source file (known to <i>make</i>), or LISP source file.
.LIST	<i>notes</i> list file.
.list	File containing a list of other files.
.ln	Library information for <i>lint</i> .
.lof	List of figures created by LaTeX.
.log	Generic log file, or a log of error messages from TeX.
.lot	List of tables created by LaTeX.
.m	Modula language source file.
.m2	Modula-2 language source file.
.make	
.mk	Makefile for <i>make</i> .
.man	Source for <i>nroff</i> or <i>troff</i> using man macros.
.me	Source for <i>nroff</i> or <i>troff</i> using me macros.

.mf	TeX metafont input file.
.ml	Gosling/Unipress Emacs Mock Lisp file.
.mm	Source for <i>nroff</i> or <i>troff</i> using mm macros.
.mon	
.month	A script that is read monthly.
.ms	Source for <i>nroff</i> or <i>troff</i> using ms macros.
.n	<i>nroff</i> source.
.NEW	
.new	New version of a file.
.nro	<i>nroff</i> source.
.O	HP64000 listing file.
.o	Relocatable object file (post-compile, pre-link); known to <i>as</i> , <i>cc</i> , <i>fc</i> , <i>pc</i> , and <i>make</i> .
.obs	Obsolete version of a file.
.OLD	
.old	Old version of a file.
.opt	File containing optional material, such as an optional part of the kernel.
.orig	Original version of a file.
.out	Standard output (and possibly standard error) from a program (for example, <i>nohup.out</i>), or an executable file output from <i>ld</i> (such as <i>a.out</i>).
.P	HP64000 cross compiled Pascal source file.
.p	Pascal language source file (known to <i>pc</i> and <i>make</i>), or PROLOG language source file.
.pag	DBM database data file.
.pi	PILOT language source file.
.pk	TeX font bitmaps in Packed Font format; denser/more recent than GF.
.prev	Previous version of a file.
.ps	PostScript files.
.pxl	TeX font bitmaps in uncompressed format; very obsolete.
.R	HP64000 relocatable file.
.r	RatFor language source file; known to <i>make</i> .
.rc	A "run commands" file, normally read when a program is invoked (for example, <i>mailx.rc</i>).
.real	Real version of a file, often one which was replaced by a front-end (for example, <i>uucico.real</i>).
.req	File containing required material, such as a required part of the kernel.
.S	HP64000 cross assembled source file.
.s	Assembler input file; known to <i>cc</i> and <i>make</i> .
.safe	
.save	Safe or saved copy of a file.
.scm	Scheme file.
.sh	Bourne shell script file; known to <i>make</i> .
.shar	Shell archive file containing output from <i>shar</i> .
.skel	Skeletal or template file.
.sl	Shared library file built by <i>ld(1)</i> ; known to <i>ld(1)</i> .
.st	File containing statistics (for example, <i>/etc/mail/sendmail.st</i>).

.sty	LaTeX style definition; should have a corresponding .doc file.
.SYSTEM	LIF Bootable by the Series 300/400 boot ROM (see Librarian chapter of <i>Pascal 3.2 Workstation System</i> , vol. 1).
.t	Text file.
.tar	File (archive) containing output from <i>tar</i> .
.tbl	Source for <i>nroff</i> table macros.
.temp	
.tmp	Temporary file.
.template	Prototype or template file.
.test	Test input or output file.
.tex	TeX source file.
.TEXT	<i>notes</i> text file, or a Pascal workstation "UCSD text format" file.
.text	
.txt	ASCII text file.
.tfm	Width information used by TeX (TeX font metrics).
.toc	Source for <i>nroff</i> table of contents macros, or table of contents created by LaTeX.
.tro	<i>troff</i> source.
.u1	
.u2	Icon intermediate code files.
.UX	HP-UX text or binary file format.
.web	Web file (Knuth's Web system).
.week	
.wk	A script that is read weekly.
.X	HP64000 absolute file.
.y	<i>yacc</i> input file; known to <i>make</i> .
.Z	File compressed by <i>compress</i> .
.z	File compressed by <i>pack</i> .
.1 .. .8	Manual entry files (sections 1 through 8), optionally followed by a letter a..z.
.<date>	File saved on given date (year, month name, YYMM, MMDD, etc.) as a snapshot of a continuously-growing logfile.
,v	RCS delta file; known to the RCS programs.

AUTHOR

suffix was developed by HP.

(ENHANCED CURSES)

NAME

term.h — terminal capabilities

SYNOPSIS

```
#include <term.h>
```

DESCRIPTION

The following data type is defined through **typedef**:

TERMINAL An opaque representation of the capabilities for a single terminal from the **terminfo** database.

The **<term.h>** header provides a declaration for the following object: *cur_term*. It represents the current terminal record from the **terminfo** database that the application has selected by calling **set_curterm()**.

The **<term.h>** header contains the variable names listed in the **Variable** column in the table in **Defined Capabilities** in **terminfo(4)**.

The following are declared as functions, and may also be defined as macros:

```
int      del_curterm(TERMINAL *oterm);
int      putp(char *const str);
int      restartterm(char *term, int fildes, int *errret);
TERMINAL * set_curterm(TERMINAL *nterm);
int      setupterm(char *term, int fildes, int *errret);
int      tgetent(char *bp, char *const name);
int      tgetflag(char id[2]);
int      tgetnum(char id[2]);
char *   tgetstr(char id[2], char **area);
char *   tgoto(char *const cap, int col, int row);
int      tigetflag(char *capname);
int      tigetnum(char *capname);
char *   tigetstr(char *capname);
char *   tparm(char *cap, long p1, long p2, long p3, long p4,
               long p5, long p6, long p7, long p8, long p9);
int      tputs(char *const str, int affcnt, int (*putfunc)(int));
```

SEE ALSO

terminfo(4), **printf()**, **putp()**, **tigetflag()**, **tgetent()**, **<curses.h>**.

CHANGE HISTORY

First released in X/Open Curses, Issue 4.

NAME

types - primitive system data types

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION**REMARKS**

The example given on this page is a typical version. The type names are in general expected to be present, although exceptions (if any) may be described in DEPENDENCIES. In most cases the fundamental type which implements each typedef is implementation dependent as long as source code which uses those typedefs need not be changed. In some cases the typedef is actually a shorthand for a commonly used type, and will not vary.

The data types defined in the include file are used in HP-UX system code; some data of these types are accessible to user code:

```
typedef struct { int r[1]; } *physadr;
typedef long daddr_t;
typedef char *caddr_t;
typedef unsigned int uint;
typedef unsigned short ushort;
typedef ushort ino_t;
typedef short cnt_t;
typedef long time_t;
typedef long dev_t;
typedef long off_t;
typedef long paddr_t;
typedef long key_t;
typedef short pid_t;
typedef long uid_t;
typedef long gid_t;
```

Note that the defined names above are standardized, but the actual type to which they are defined may vary between HP-UX implementations.

The meanings of the types are:

<i>physadr</i>	used as a pointer to memory; the pointer is aligned to follow hardware-dependent instruction addressing conventions.
<i>daddr_t</i>	used for disk addresses except in an inode on disk, see <i>fs(4)</i> .
<i>caddr_t</i>	used as an untyped pointer or a pointer to untyped memory.
<i>uint</i>	shorthand for <i>unsigned integer</i> .
<i>ushort</i>	shorthand for <i>unsigned short</i> .
<i>ino_t</i>	used to specify I-numbers.
<i>cnt_t</i>	used in some implementations to hold reference counts for some kernel data structures.
<i>time_t</i>	time encoded in seconds since 00:00:00 GMT, January 1, 1970.
<i>dev_t</i>	specifies kind and unit number of a device, encoded in two parts known as major and minor.
<i>off_t</i>	offsets measured in bytes from the beginning of a file.
<i>paddr_t</i>	used as an integer type which is properly sized to hold a pointer.
<i>key_t</i>	the type of a key used to obtain a message queue, semaphore, or shared memory identifier, see <i>stdipc(3C)</i> .
<i>pid_t</i>	used to specify process and process group identifiers.
<i>uid_t</i>	used to specify user identifiers.
<i>gid_t</i>	user to specify group identifiers.

SEE ALSO

fs(4), stdipc(3C).

STANDARDS CONFORMANCE

<sys/types.h>: AES, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1


t

NAME

unctrl.h - definitions for unctrl()

DESCRIPTION

The **<unctrl.h>** header defines the **chtype** type as defined in **< curses.h>**.

The following is declared as a function, and may also be defined as a macro:

```
char *unctrl(chtype c);
```

SEE ALSO

< curses.h>.

CHANGE HISTORY

First released in X/Open Curses, Issue 4.

NAME

unitsd: unistd.h - standard structures and symbolic constants

SYNOPSIS

```
#include <unistd.h>
```

DESCRIPTION

The header **<unistd.h>** defines the following structures and symbolic constants:

Symbolic constants for the **access()** function:

R_OK	Test for read permission.
W_OK	Test for write permission.
X_OK	Test for execute (search) permission.
F_OK	Test for existence of file.

The constants **F_OK**, **R_OK**, **W_OK**, and **X_OK** and the expressions **R_OK|W_OK**, **R_OK|X_OK**, and **R_OK|W_OK|X_OK** all have distinct values.

Symbolic constant representing a null pointer:

NULL

Symbolic constants for the **lseek()** and **fcntl()** functions (the following constants have distinct values):

SEEK_SET	Set file offset to "offset".
SEEK_CUR	Set file offset to current plus "offset".
SEEK_END	Set file offset to EOF plus "offset".

Symbolic constants (with fixed values):

_POSIX_VERSION	Integer value indicating version of <i>IEEE Std1003.1</i> standard implemented. The current value is 199009L, indicating the (4-digit) year and (2-digit) month that the standard was approved by the IEEE Standards Board. However, if any of the symbols _AES_SOURCE , _XPG3 , or _POSIX1_1988 is defined before <unistd.h> is included, the value of this symbol will be 198808L.
_POSIX2_VERSION	Integer value indicating version of <i>IEEE Std1003.2</i> standard implemented. The current value is 199209L, indicating the (4-digit) year and (2-digit) month that the standard was approved by the IEEE Standards Board.
_POSIX2_C_VERSION	Integer value indicating version of <i>IEEE Std1003.2</i> C-Language Binding Option implemented. The current value is 199209L, indicating the (4-digit) year and (2-digit) month that the standard was approved by the IEEE Standards Board.
_XOPEN_VERSION	Integer value indicating issue number of the X/Open Portability Guide implemented. The current value is 4, indicating Issue 4. However, if the symbol _XPG3 is defined before <unistd.h> is included, the value of this symbol will be 3.

The following symbolic constants are defined in this header if the state of the corresponding option or restriction does not vary after compilation. If a symbol is absent from this header, the value or presence of the corresponding option or restriction should be determined at execution time through **sysconf()** or **pathconf()**:

_POSIX_CHOWN_RESTRICTED	The use of chown() is restricted to processes that have appropriate privileges.
_POSIX_JOB_CONTROL	Implementation supports job control (true of all HP-UX implementations).
_POSIX_NO_TRUNC	Pathname components longer than NAME_MAX generate an error.
_POSIX_REALTIME_SIGNALS	Implementation supports Realtime Signals Extensions (true of all HP-UX implementations).

<code>_POSIX_SAVED_IDS</code>	Effective user and group are saved across an <code>exec()</code> call (true of all HP-UX implementations).
<code>_POSIX_FSYNC</code>	Implementation supports File Synchronization (true of all HP-UX implementations). See <code>open(2)</code> .
<code>_POSIX_SYNCHRONIZED_IO</code>	Implementation supports Synchronized IO (true of all HP-UX implementations). See <code>open(2)</code> .
<code>_POSIX_VDISABLE</code>	Terminal special characters can be disabled using this character (see <code>termio(7)</code>).
<code>_POSIX_THREADS</code>	Implementation supports POSIX threads.
<code>_POSIX2_C_BIND</code>	All POSIX.2 C-language functionality is provided in the default libraries used by the <code>c89</code> C compiler (see <code>cc(1)</code>).
<code>_POSIX2_LOCALEDEF</code>	New locales can be defined by using the <code>localedef</code> command (see <code>localedef(1M)</code>).
<code>_POSIX2_UPE</code>	The system supports <i>IEEE Std1003.2a</i> (POSIX User Portability Utilities Option).
<code>_POSIX2_CHAR_TERM</code>	At least one terminal exists that supports all required POSIX.2a commands.

All symbolic constants whose names begin `_CS`, `_PC`, and `_SC` (see `confstr(3C)`, `pathconf(2)`, and `sysconf(2)`) are defined.

The following symbolic constants for file streams are defined:

<code>STDIN_FILENO</code>	File number of standard input (<i>stdin</i>).
<code>STDOUT_FILENO</code>	File number of standard output (<i>stdout</i>).
<code>STDERR_FILENO</code>	File number of standard error (<i>stderr</i>).

The types `size_t`, `ssize_t`, `uid_t`, `gid_t`, `off_t`, and `pid_t` are defined.

Declarations are provided for the following functions:

<code>access()</code>	<code>alarm()</code>	<code>brk()</code>	<code>chdir()</code>
<code>chown()</code>	<code>chroot()</code>	<code>close()</code>	<code>confstr()</code>
<code>crypt()</code>	<code>ctermid()</code>	<code>cuserid()</code>	<code>dup()</code>
<code>dup2()</code>	<code>encrypt()</code>	<code>endusershell()</code>	<code>exec1()</code>
<code>execle()</code>	<code>execlp()</code>	<code>execv()</code>	<code>execve()</code>
<code>execvp()</code>	<code>_exit()</code>	<code>fchown()</code>	<code>fork()</code>
<code>fpathconf()</code>	<code>fsync()</code>	<code>ftruncate()</code>	<code>getcwd()</code>
<code>getegid()</code>	<code>geteuid()</code>	<code>getgid()</code>	<code>getgroups()</code>
<code>gethostname()</code>	<code>getlogin()</code>	<code>getopt()</code>	
<code>getpass()</code>	<code>getpgrp()</code>	<code>getpgrp2()</code>	<code>getpid()</code>
<code>getppid()</code>	<code>getuid()</code>	<code>getusershell()</code>	<code>initgroups()</code>
<code>ioctl()</code>	<code>isatty()</code>	<code>link()</code>	<code>lockf()</code>
<code>logname()</code>	<code>lseek()</code>	<code>mkstemp()</code>	<code>mktemp()</code>
<code>nice()</code>	<code>pathconf()</code>	<code>pause()</code>	<code>pipe()</code>
<code>prealloc()</code>	<code>read()</code>	<code>readlink()</code>	<code>rmdir()</code>
<code>sbrk()</code>	<code>setgid()</code>	<code>setgroups()</code>	<code>sethostname()</code>
<code>setpgid()</code>	<code>setpgrp()</code>	<code>setpgrp2()</code>	<code>setresgid()</code>
<code>setresuid()</code>	<code>setsid()</code>	<code>setuid()</code>	<code>setusershell()</code>
<code>sgetl()</code>	<code>sleep()</code>	<code>sputl()</code>	<code>swab()</code>
<code>swapon()</code>	<code>symlink()</code>	<code>sync()</code>	<code>sysconf()</code>
<code>tcgetpgrp()</code>	<code>tcsetpgrp()</code>	<code>truncate()</code>	<code>ttyname()</code>
<code>ttyslot()</code>	<code>unlink()</code>	<code>vfork()</code>	<code>write()</code>

SEE ALSO

`access(2)`, `chown(2)`, `confstr(3C)`, `exit(2)`, `fcntl(2)`, `kill(2)`, `lseek(2)`, `open(2)`, `pathconf(2)`, `sysconf(2)`, `limits(5)`, `stdsyms(5)`, `termio(7)`.

AUTHOR

`unistd` was developed by HP.

STANDARDS CONFORMANCE

`<unistd.h>`: AES, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1, POSIX.2, POSIX.4

NAME

values - machine-dependent values

SYNOPSIS

#include <values.h>

DESCRIPTION

This file contains a set of manifest constants, conditionally defined for particular processor architectures.

The model assumed for integers is binary representation (one's or two's complement), where the sign is represented by the value of the high-order bit.

BITS(<i>type</i>)	The number of bits in a specified type (e.g., int).
HIBITS	The value of a short integer with only the high-order bit set (in most implementations, 0x8000).
HIBITL	The value of a long integer with only the high-order bit set (in most implementations, 0x80000000).
HIBITI	The value of a regular integer with only the high-order bit set (usually the same as HIBITS or HIBITL).
MAXSHORT	The maximum value of a signed short integer (in most implementations, 0x7FFF \equiv 32767).
MAXLONG	The maximum value of a signed long integer (in most implementations, 0x7FFFFFFF \equiv 2147483647).
MAXINT	The maximum value of a signed regular integer (usually the same as MAXSHORT or MAXLONG).
MAXFLOAT, LN_MAXFLOAT	The maximum value of a single-precision floating-point number, and its natural logarithm.
MAXDOUBLE, LN_MAXDOUBLE	The maximum value of a double-precision floating-point number, and its natural logarithm.
MINFLOAT, LN_MINFLOAT	The minimum positive value of a single-precision floating-point number, and its natural logarithm.
MINDOUBLE, LN_MINDOUBLE	The minimum positive value of a double-precision floating-point number, and its natural logarithm.
FSIGNIF	The number of significant bits in the mantissa of a single-precision floating-point number.
DSIGNIF	The number of significant bits in the mantissa of a double-precision floating-point number.

FILES

/usr/include/values.h

SEE ALSO

intro(3), math(5).

STANDARDS CONFORMANCE

<values.h>: XPG2

NAME

varargs - handle variable argument list

SYNOPSIS

```
#include <varargs.h>
va_alist
va_dcl
void va_start(pvar)
va_list pvar;
type va_arg(pvar, type)
va_list pvar;
void va_end(pvar)
va_list pvar;
```

DESCRIPTION

This set of macros enables programmers to write portable procedures that accept variable argument lists. Routines that have variable argument lists (such as `printf()`) but do not use `varargs` are inherently nonportable, because different machines use different argument-passing conventions (see *printf(3S)*).

`va_alist` is used as the parameter list in a function header.

`va_dcl` is a declaration for `va_alist`. No semicolon should follow `va_dcl`.

`va_list` is a type defined for the variable used to traverse the list.

`va_start` is called to initialize `pvar` to the beginning of the list.

`va_arg` returns the next argument in the list pointed to by `pvar`. `type` is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, because it cannot be determined at runtime.

`va_end` is used to clean up.

Multiple traversals, each bracketed by `va_start` ... `va_end`, are possible.

EXAMPLE

The following example shows a possible implementation of `execl()` (see *exec(2)*):

```
#include <varargs.h>
#define MAXARGS 100
/* execl is called by
   execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
    va_list ap;
    char *file;
    char *args[MAXARGS];
    int argno = 0;

    va_start(ap);
    file = va_arg(ap, char *);
    while ((args[argno++] = va_arg(ap, char *)) != (char *)0);
    va_end(ap);
    return execv(file, args);
}
```

The next example illustrates how a function that receives variable arguments can pass these arguments down to other functions. To accomplish this, the first routine (`log_errors()` in this example) which receives the variable argument list must pass the address pointer resulting from a call to `va_start()` on to any subsequent calls that need to access this same variable argument list. All routines that receive this address pointer (`v_print_log()` in this example) need only to use `va_arg()` to access the original variable argument list just as if they were the original routine to be passed the variable arguments.

In this example, one can imagine that there are a series of other routines (such as a `log_warning()` and `log_message()`) that also call the `v_print_log()` function.

```
#include <stdio.h>
#include <varargs.h>
#include <unistd.h>

int error_count;

/* VARARGS4 -- for lint */
int
log_errors(log_fp, func_name, err_num, msg_fmt, va_alist)
FILE *log_fp;
char *func_name;
int err_num;
char *msg_fmt;
va_dcl
{
    va_list ap;

    /* Print error header information */
    (void) fprintf(log_fp, "\nERROR in process %d\n", getpid());
    (void) fprintf(log_fp, " function \"%s\": ", func_name);
    switch(err_num)
    {
        case ILLEGAL_OPTION:
            (void) fprintf(log_fp, "illegal option\n");
            break;
        case CANNOT_PARSE:
            (void) fprintf(log_fp, "cannot parse input file\n");
            break;
        ...
    }

    /*
     * Get pointer to first variable argument so that we can
     * pass it on to v_print_log(). We do this so that
     * v_print_log() can access the variable arguments passed
     * to this routine.
     */
    va_start(ap);

    v_print_log(log_fp, msg_fmt, ap);

    va_end(ap);
}

/* VARARGS2 -- for lint */
int
v_print_log(log_fp, fmt, ap)
FILE *log_fp;
char *fmt;
va_list ap;
{
    /*
     * If "%Y" is the first two characters in the format string,
     * a second file pointer has been passed in to print general
     * message information to. The rest of the format string is
     * a standard printf(3S) format string.
     */
}
```



```

if ((*fmt == '%') && (*(fmt + 1) == 'Y'))
{
    FILE *other_fp;

    fmt += 2;

    other_fp = (FILE *) va_arg(ap, char *);
    if (other_fp != (FILE *) NULL)
    {
        /*
         * Print general message information to additional stream.
         */
        (void) vfprintf(other_fp, fmt, ap);
        (void) fflush(other_fp);
    }
}

/*
 * Now print it to the log file.
 */
(void) vfprintf(log_fp, fmt, ap);
}

```

WARNINGS

It is up to the calling routine to specify how many arguments there are, because it is not always possible to determine this from the stack frame. For example, `exec1()` is passed a zero pointer to signal the end of the list. `printf()` can determine how many arguments are present by the format.

It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, because arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int`, and converts `float` arguments to `double`, before passing them to a function.

SEE ALSO

`exec(2)`, `vprintf(3S)`.

STANDARDS CONFORMANCE

va_alist: AES, SVID2, SVID3, XPG2, XPG3, XPG4

va_arg: SVID2, SVID3, XPG2, XPG3, XPG4

va_dcl: SVID2, SVID3, XPG2, XPG3, XPG4

va_end: SVID2, SVID3, XPG2, XPG3, XPG4

va_list: SVID2, SVID3, XPG2, XPG3, XPG4

va_start: SVID2, SVID3, XPG2, XPG3, XPG4

<varargs.h>: AES, SVID3, XPG2, XPG3, XPG4

NAME

x_open - pointer manual entry for X/Open Conformance Statement Questionnaire

DESCRIPTION

This entry is a pointer entry for accessing the X/Open Conformance Statement Questionnaire for HP 9000 Series 700/800 HP-UX systems. To access the conformance statement, use the command:

man x_open_800

FILES

/usr/share/man/man5.Z/x_open_800.5

Series 700/800 X/Open Conformance Statement Questionnaire (preformatted and compressed)

Section 7

Device (Special) Files

Section 7

Device (Special) Files

NAME

intro - introduction to device special files

DESCRIPTION

This section describes the device special files used to access HP peripherals and device drivers. The names of the entries are generally derived from the type of device being described (disk, terminal, etc.), not the names of the device special files or device drivers themselves. Characteristics of both the hardware device and the corresponding HP-UX device driver are discussed where applicable.

The devices can be classified in two categories, **raw** and **block**. A raw or character-mode device, such as a line printer, transfers data in an unbuffered stream and uses a character device special file.

Block devices, as the name implies, transfer data in blocks by means of the system's normal buffering mechanism. Block devices use block device special files and may have a character device interface too.

A device special file name becomes associated with a device when the file is created, using the *mksf*(1M), *insf*(1M), or *mknod*(1M) commands. When creating device special files, it is recommended that the following standard naming convention be used:

/dev/prefix/devspec[options]

prefix indicates the subdirectory for the device class (for example, **rdsk** for raw device special files for disks, **dsk** for block device special files for disks, **rmt** for raw tape devices).

devspec indicates hardware path information and is typically in the format **c#t#d#** as follows:

c# Instance number assigned by the operating system to the interface card. There is no direct correlation between instance number and physical slot number.

t# Target address on a remote bus (for example, SCSI or HP-IB address).

d# Device unit number at the target address (for example, SCSI LUN).

options Further qualifiers, such as disk section **s#** (for backward compatibility), tape density selection for a tape device, or surface specification for magneto-optical media.

Hardware path information can be derived from *ioscan*(1M) output.

EXAMPLES

The following is an example of a disk device special file name:

/dev/dsk/c0t6d0

where **dsk** indicates block disk access and **c0t6d0** indicates disk access at interface card instance 0, target address 6, and unit 0. Absence of **s#** indicates access to the entire disk (see *disk*(7) for details).

The following is an example of a tape device special file name:

/dev/rmt/c2t3d0QIC150

where **rmt** indicates raw magnetic tape, **c2** indicates that the device is connected to interface card instance 2, **t3** indicates that target device address is set to 3, **d0** indicates that the tape transport resides at unit address 0, and **QIC150** identifies the tape format as QIC150 (see *mt*(7) for details).

WARNINGS

In the past, other naming conventions have been used for device special files. Using *ln*(1) to create a link between the old and new standard name is useful as a temporary expedient until all programs using an old naming convention have been converted.

SEE ALSO

hier(5), *ioscan*(1M), *mksf*(1M), *insf*(1M), *lssf*(1M).

The system administrator manual for your system.

NAME

arp - Address Resolution Protocol

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and hardware station addresses. It is used by all LAN drivers.

ARP caches Internet-to-hardware station address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message that requires the mapping, and broadcasts a message on the associated network requesting the address mapping if the **ether** encapsulation method has been enabled for the interface. If a response is provided, the new mapping is cached and any pending message is transmitted. ARP queues at most one packet while waiting for a mapping request to be responded to; only the most recently "transmitted" packet is kept.

To facilitate communications with systems that do not use ARP, **ioctl** calls are provided to enter and delete entries in the Internet-to-hardware station address tables.

Application Usage:

```
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/if_ether.h>
struct arpreq arpreq;

ioctl(s, SIOCSARP, (caddr_t)&arpreq);
ioctl(s, SIOCGARP, (caddr_t)&arpreq);
ioctl(s, SIOCDEARP, (caddr_t)&arpreq);
```

Each **ioctl** call takes the same structure as an argument. **SIOCSARP** sets an ARP entry, **SIOCGARP** gets an ARP entry, and **SIOCDEARP** deletes an ARP entry. These **ioctl** calls can be applied to any socket descriptor *s*, but only by the super-user. The **arpreq** structure contains:

```
/*
 * ARP ioctl request
 */
struct arpreq {
    int32_t ifindex;
    int32_t arp_flags;           /* flags */
    int32_t arp_hw_addr_len;    /* hardware address length */
    struct sockaddr arp_pa;     /* protocol address */
    struct sockaddr arp_ha;     /* hardware address */
    u_char  arp_pad[242];      /* buffer for link specific info. */
};
/* arp_flags field values */
#define ATF_COM          0x02    /* ARP on ether */
#define ATF_PERM         0x04    /* permanent entry */
#define ATF_PUBL         0x08    /* publish entry */
#define ATF_SNAPFDDI     0x200   /* SNAP - FDDI */
#define ATF_SNAP8025     0x400   /* SNAP - 8025 */
#define ATF_IEEE8025     0x800   /* IEEE - 8025 */
#define ATF_FCSNAP       0x4000  /* Fibre Channel SNAP */
```

The address family for the *arp_pa* **sockaddr** must be **AF_INET**; for the *arp_ha* **sockaddr** it must be **AF_UNSPEC**. The only flag bits that can be written are **ATF_PERM**, and **ATF_PUBL**. Fibre Channel hosts only support the **ATF_PERM** flag. **ATF_PERM** causes the entry to be permanent. **ATF_PUBL** specifies that the ARP code should respond to ARP requests for the indicated host coming from other machines. This allows a host to act as an *ARP server*, which may be useful in convincing an ARP-only machine to talk to a non-ARP machine.

ARP watches passively for hosts impersonating the local host (i.e., a host that responds to an ARP mapping request for the local host's address).

DIAGNOSTICS

duplicate IP address!! sent from ethernet address: %x:%x:%x:%x:%x:%x.

This message printed on the console screen means that ARP has discovered another host on the local network that responds to mapping requests for its own Internet address.

WARNINGS

To enable the **ether** encapsulation method, use the **ifconfig** command (see *ifconfig(1M)*).

AUTHOR

ARP was developed by the University of California, Berkeley.

SEE ALSO

ifconfig(1M), *inet(3N)*, *lan(7)*, *arp(1M)*.

An Ethernet Address Resolution Protocol, RFC826, Dave Plummer, Network Information Center, SRI.


a

NAME

autochanger - SCSI interfaces for medium changer device and magneto-optical autochanger surface device

DESCRIPTION

An autochanger is a SCSI mass storage device, consisting of a mechanical changer device, one or more data transfer devices (such as optical disk drives), and media (such as optical disks) for data storage. The mechanical changer moves media between storage and usage locations within the autochanger.

Depending on system architecture, one of two medium changer drivers (**schgr** or **autox0**) provides access to the medium changer device; a module (**ssrfc**) provides access to the surfaces of the optical disks.

Two levels of functionality are provided by the medium changer drivers. The mechanical changer device can be accessed directly to move media within the autochanger. Alternatively, media surfaces can be accessed as unique devices, causing the changer driver to move the media into a drive to perform an I/O request.

The **schgr** and **autox0** medium changer device drivers follow the SCSI specification for medium changer devices to provide a generic medium changer interface, making it feasible to construct an application level driver for any mechanical changer, jukebox, library, or autochanger device (MO, tape, CD-ROM).

However, the **ssrfc** module is provided specifically to support Hewlett-Packard magneto-optical disk autochanger products.

Device Naming Convention

The device naming convention for the autochanger driver enables accessing the changer device, as well as individual media surfaces. Block devices for autochangers reside in **/dev/ac**, character devices reside in **/dev/rac**. Within these directories, names are derived from the "c#t#d#" device naming convention (explained in *intro(7)*), with the surface descriptor appended at the end. Unique device names are determined by the card instance, target address of the SCSI changer device, LUN of the SCSI changer device, and the surface descriptor.

The surface descriptor can be zero or non-specified for the changer device. Also, there is no block special file for the changer itself. For example,

/dev/rac/clt5d0

is the character special file for the changer at SCSI target address 5 and LUN 0, attached to SCSI card instance 1, and is equivalent to **/dev/rac/clt5d0_0**.

Any given surface is described by the card instance, SCSI target address and SCSI LUN of the changer, and then appended with a surface descriptor for the slot number and side. For example,

/dev/ac/clt5d0_1a

is the block special file for surface 1a of the autochanger just mentioned and

/dev/rac/clt5d0_1a

is the character special file for the same surface 1a.

Major and Minor Number Descriptions

The following shows the bit assignments (**dev_t** format) used by the changer drivers to access the changer device and each surface within an autochanger:

0-7	8-15	16-19	20-22	23-31
MAJOR	INSTANCE	TARGET	LUN	SURFACE

MAJOR is the major number of the appropriate driver, INSTANCE is the card instance of the SCSI interface to which the changer device is attached, TARGET is the SCSI target address of the changer device, LUN is the SCSI LUN of the changer device, and SURFACE is the unique descriptor of each surface in the autochanger, as described in the following table. (Note, the surface descriptors refer to bits 23-31.)

Surface	Surface Descriptor
changer device	0
1a	01
1b	02
2a	03
2b ...	04 ...
31b	3e
32a	3f
32b	40

All fields in the device number are specified in hexadecimal notation. Note that there is no support for hard partitions (sections) in this minor number. If desired, partitioning can be achieved via LVM soft-partitioning schemes.

The major numbers used by the changer drivers are:

	b_major	c_major
schgr	29	231
autox0	30	230

Following are long listings showing the major and minor numbers associated with the device special file names of the first surface and the changer:

schgr:

```
brw-rw-rw- 1 root sys 29 0x015001 Apr 22 10:22 /dev/ac/clt5d0_1a
crw-rw-rw- 1 root sys 231 0x015001 Apr 22 10:22 /dev/rac/clt5d0_1a
crw-rw-rw- 1 root sys 231 0x015000 Apr 22 10:22 /dev/rac/clt5d0
```

autox0:

```
brw-rw-rw- 1 root sys 30 0x015001 Apr 24 11:35 /dev/ac/clt5d0_1a
crw-rw-rw- 1 root sys 230 0x015001 Apr 24 11:35 /dev/rac/clt5d0_1a
crw-rw-rw- 1 root sys 230 0x015000 Apr 24 11:35 /dev/rac/clt5d0
```

MAGNETO-OPTICAL AUTOCHANGER SURFACE DEVICE ACCESS

To access disk surfaces within HP magneto-optical libraries, it is necessary to include the entry for the surface module, **ssrfc**, in the system configuration file **/stand/system**, as well as an entry for the appropriate SCSI changer driver, **schgr** or **autox0**, depending on architecture. The **ssrfc** module enables accessing a magneto-optical disk surface much like a disk device. The disk is moved into an idle drive by the changer, then the requested disk I/O operation is performed. Upon completion of the request, the disk is returned to its storage location within the autochanger.

The surface module allows concurrent access to as many disks as there are drives in the autochanger product. Requests for I/O on additional disks within the autochanger are blocked awaiting an available drive resource.

By default, some commands (such as **mount**, **newfs**, and **mediainit**) open the device with the **O_NDELAY** flag set. Invocations of these commands on an autochanger surface do not wait for a drive resource to become available. Instead, these requests return with **EBUSY** if no drive is available.

Developers using the surface module functionality to access autochanger disks can invoke the open system call with the **O_NDELAY** flag to achieve this same "non-blocking" behavior:

```
error = open("/dev/rac/clt5d0_1a",O_RDWR | O_NDELAY);
```

If it is acceptable to block waiting for an available drive resource, the **O_NDELAY** flag is unnecessary.

Here is a sample script to access multiple disk surfaces in an autochanger that has 2 drives, minimizing blocking:

```
dd if=/dev/rdsk/c0t0d0 of=/dev/rac/clt5d0_1a bs=64k &
dd if=/dev/rdsk/c0t1d0 of=/dev/rac/clt5d0_2a bs=64k &
wait
dd if=/dev/rdsk/c0t2d0 of=/dev/rac/clt5d0_1b bs=64k &
dd if=/dev/rdsk/c0t3d0 of=/dev/rac/clt5d0_2b bs=64k &
wait
```



...

For developers, the `ioctl` functions available for accessing magneto-optical disk surfaces are described in the manual pages for SCSI disk drivers. Several `ioctl` functions provided specifically for magneto-optical disks will be described here briefly. Included from `<sys/scsi.h>`:

a

```
#define SIOC_WRITE_WOE      _IOW('S', 17, int)
#define SIOC_VERIFY_WRITES _IOW('S', 18, int)
#define SIOC_ERASE          _IOW('S', 19, struct scsi_erase)
#define SIOC_VERIFY_BLANK   _IOW('S', 20, struct scsi_verify)
#define SIOC_VERIFY         _IOW('S', 21, struct scsi_verify)
```

`SIOC_ERASE` (erase) and `SIOC_WRITE_WOE` (write without erase) can be used together on character special devices. By performing a pre-erase pass of magneto-optical disks, then later setting the SCSI disk driver in write-without-erase mode, improved write performance can be achieved, eliminating the two-pass erase-then-write which is normally necessary on magneto-optical devices.

`SIOC_VERIFY_WRITES` (write and verify) performs a verification pass on any writes to magneto-optical disks. This is a good safeguard for data integrity. However, write operations performed with the verification pass exhibits a decrease in performance. When used with pre-erase and write-without-erase, write and verify provide increased reliability of data without decreased performance. HP recommends operating in write-and-verify mode if also performing write-without-erase.

The following are additional `ioctl` functions that might be desirable for some magneto-optical products, included from `<sys/scsi.h>`:

```
#define SIOC_GET_IR        _IOR('S', 14, int)
#define SIOC_SET_IR        _IOW('S', 15, int)
#define SIOC_SYNC_CACHE    _IOW('S', 70, int)
```

`SIOC_GET_IR` determines the current state of immediate reporting (write caching) on the device. `SIOC_SET_IR` enables or disables immediate reporting on the device. If `SIOC_SET_IR` is used to enable write caching, it may be desirable to flush the write cache using the `SIOC_SYNC_CACHE` `ioctl` function. The command `/usr/sbin/scsictl` may be used to perform the pre-erase of magneto-optical disks, set and check the status of immediate reporting.

With the surface module configured, several `ioctl` functions to get status and information from the changer device are also available. These are `SIOC_ELEMENT_ADDRESSES`, `SIOC_ELEMENT_STATUS`, and `SIOC_INQUIRY`; they are explained further in the following section on the changer driver. Functions that modify the state of the autochanger are not allowed when the surface module is configured into the kernel.

SCSI MEDIUM CHANGER DEVICE DRIVER

The SCSI medium changer device driver performs moves between different media locations within an autochanger. Each potential media location has a specific element address and is one of the following element types:

<i>storage</i>	A location to hold a unit of media not currently in use. Typically most media will be located in this type of element.
<i>import/export</i>	A location for inserting and removing media from the device. Movement of a unit of media to this type of location is in effect an eject operation. Movement of a unit of media from this type of location is a load operation.
<i>data transfer</i>	A location for accessing media data. This is generally the location of a device that reads and/or writes data on the media being handled by the media changer device. Movement to this type of location is a physical-media-mount operation. Movement from this type of location is a physical-media-unmount operation.
<i>media transport</i>	A location for media movement. Media is generally temporarily located in this type of element only during actual media movement.

Changer Control Requests

The following `ioctl` functions are included from `<sys/chgrio.h>`:

```
#define CHGR_SSRFC_IS_PRESENT _IOR('X', 1, int)
#define CHGR_CLEAR_RESET     _IO('X', 2)
```

CHGR_SSRFC_IS_PRESENT

For developers. To determine if the surface module functionality (**ssrfc**) is currently configured in the kernel.

CHGR_CLEAR_RESET

For developers. autox0 driver only. To clear a powerfail recovery condition in the SCSI changer driver. The **CHGR_CLEAR_RESET** ioctl function will be necessary for developers using the SCSI changer driver (autox0) to move media within the medium changer, but not using the surface module for transparent access to magneto-optical disks. In the event of an **ECONNRESET** error return from any changer ioctl call, a **CHGR_CLEAR_RESET** call will be necessary prior to any further media moves. This alerts the application of a possible power failure, and allows the developer an opportunity to reset data structures, and re-reserve elements in the medium changer, prior to further operations.

The following ioctl functions and structure definitions are included from **<sys/scsi.h>**:

```
#define SIOC_INIT_ELEM_STAT      _IO('S', 51)
#define SIOC_ELEMENT_ADDRESSES  _IOW('S', 52, struct element_addresses)
#define SIOC_ELEMENT_STATUS      _IOWR('S', 53, struct element_status)
#define SIOC_RESERVE             _IOW('S', 54, struct reservation_parms)
#define SIOC_RELEASE             _IOW('S', 55, struct reservation_parms)
#define SIOC_MOVE_MEDIUM        _IOW('S', 56, struct move_medium_parms)
#define SIOC_EXCHANGE_MEDIUM    _IOW('S', 57, struct exchange_medium_parms)

/* structure for SIOC_ELEMENT_ADDRESSES ioctl */
struct element_addresses {
    unsigned short  first_transport;
    unsigned short  num_transports;
    unsigned short  first_storage;
    unsigned short  num_storages;
    unsigned short  first_import_export;
    unsigned short  num_import_exports;
    unsigned short  first_data_transfer;
    unsigned short  num_data_transfers;
};

/* structure for SIOC_ELEMENT_STATUS ioctl */
struct element_status {
    unsigned short element;           /* element address */

    unsigned int  resv1:2;
    unsigned int  import_enable:1; /* allows media insertion (load) */
    unsigned int  export_enable:1; /* allows media removal (eject) */
    unsigned int  access:1;        /* transport element accessible */
    unsigned int  except:1;        /* is in an abnormal state */
    unsigned int  operatr:1;       /* medium positioned by operator */
    unsigned int  full:1;          /* holds a a unit of media */

    unsigned char resv2;
    unsigned char sense_code;      /* info. about abnormal state */
    unsigned char sense_qualifier; /* info. about abnormal state */

    unsigned int  not_bus:1;       /* transfer device SCSI bus differs */
    unsigned int  resv3:1;
    unsigned int  id_valid:1;      /* bus_address is valid */
    unsigned int  lu_valid:1;      /* lun is valid */
    unsigned int  sublu_valid:1;   /* sub_lun is valid */
    unsigned int  lun:3;          /* transfer device SCSI LUN */

    unsigned char bus_address;     /* transfer device SCSI address */
    unsigned char sub_lun;         /* sub-logical unit number */

    unsigned int  source_valid:1; /* source_element is valid */
    unsigned int  invert:1;       /* media in element was inverted */
    unsigned int  resv4:6;
};
```

a

```

        unsigned short source_element; /* last storage medium location */
        char          pri_vol_tag[36]; /* volume tag (device optional) */
        char          alt_vol_tag[36]; /* volume tag (device optional) */
        unsigned char misc_bytes[168]; /* device specific */
    };

    /* structure for SIOC_RESERVE and SIOC_RELEASE ioctls */
    struct reservation_parms {
        unsigned short element;
        unsigned char  identification;
        unsigned char  all_elements;
    };

    /* structure for SIOC_MOVE_MEDIUM ioctl */
    struct move_medium_parms {
        unsigned short transport;
        unsigned short source;
        unsigned short destination;
        unsigned char  invert;
    };

    /* structure for SIOC_EXCHANGE_MEDIUM ioctl */
    struct exchange_medium_parms {
        unsigned short transport;
        unsigned short source;
        unsigned short first_destination;
        unsigned short second_destination;
        unsigned char  invert_first;
        unsigned char  invert_second;
    };

```

SIOC_INIT_ELEM_STAT

Cause the media changer device to take inventory. As a result, the media changer device determines the status of each and every element address, including the presence or absence of a unit of media. This is a mechanical operation which can take time. This function only necessary in the event of a severe error of the media changer. If using the surface module (ssrfc) to move disks, this level of error recovery is handled within the surface module.

SIOC_ELEMENT_ADDRESSES

Determine the element addresses supported by a media changer device. The first valid element address and the number of elements is indicated for each element type. These element addresses may be used as source and destination location arguments.

SIOC_ELEMENT_STATUS

Determine the status of an element. The element address for which status information is requested is specified via the **element** field. The resulting status data indicates the presence or absence of a unit of media in that element address as well as other information about the element address.

SIOC_RESERVE and SIOC_RELEASE

Control access to element addresses. Depending on the device, reservations may limit operator control of those element addresses in the media changer device. Specific element addresses can be reserved to handle interlocking between multiple requesters if each requester has a unique reservation identification. The value zero in the **all_elements** field specifies that a single element address should be reserved or released. An element address reserved in this manner can not be reserved by another single element address reservation using a different reservation identification. The **reservation** field specifies the reservation identification. The **element** field specifies the element address to be reserved.

The value "1" in the **all_elements** field indicates that all element addresses should be reserved. The **reservation** and **element** fields should contain the value zero since these fields are not meaningful when reserving all element addresses. Reserving all element addresses is primarily useful for limiting operator control.

SIOC_MOVE_MEDIUM and SIOC_EXCHANGE_MEDIUM

Reposition unit(s) of media. Depending on the source and destination element types, this may result in a media load, eject, or simple repositioning. Media can be "flipped" using values of "1" in the **invert**, **invert_first**, or **invert_second** fields. The **SIOC_EXCHANGE_MEDIUM** ioctl

repositions two different units of media. One unit of media is moved from the element specified by the **source** field to the element specified by the **first_destination** field. A second unit of media is moved from the element specified by the **first_destination** field to the element specified by the **second_destination** field. In an autochanger with multiple changer mechanisms, or a media staging area, an exchange occurs if the **source** and **second_destination** fields are the same.

DEPENDENCIES

To obtain access to disk surfaces within HP magneto-optical libraries, the **ssrfc** module must be specified in the system configuration file. The **ssrfc** module depends on either the **schgr** driver, or the **autox0** driver. If **ssrfc** is to be included, then one or both of **schgr** or **autox0** must also be included.

DEFAULT CONFIGURATIONS

By default, **ssrfc**, **schgr**, and **autox0** are not included in the system configuration (**/stand/system**) file.

EXAMPLES

The following example uses the **SIOC_ELEMENT_ADDRESSES** and **SIOC_ELEMENT_STATUS** **ioctl** functions to get bus address information about the drives in an HP magneto-optical autochanger:

```
int                last_drive_el;
struct element_addresses  el_addrs;
struct element_status    el_stat;

/*
 * Changer attached to card instance 1, with SCSI target id 5, lun 0.
 */
fd = open("/dev/rac/clt5d0", O_RDWR);
if ((error = ioctl(fd, SIOC_ELEMENT_ADDRESSES, &el_addrs)) != 0) {
    syserr("ioctl: SIOC_ELEMENT_ADDRESSES");
    return -1;
} else {
    last_drive_el = el_addrs.first_data_transfer
        + el_addrs.num_data_transfers - 1;
    for (i = el_addrs.first_data_transfer; i <= last_drive_el; i++) {
        el_stat.element = i;
        if ((error = ioctl(fd, SIOC_ELEMENT_STATUS, &el_stat)) != 0) {
            syserr("ioctl: SIOC_ELEMENT_ADDRESSES");
            return -1;
        } else {
            /*
             * You may wish to also check some of the other fields
             * in the el_stat structure to verify that the data is
             * valid. Fields: el_stat.access (ac accessible),
             * el_stat.except (exception).
             */
            if (! el_stat.not_bus && el_stat.id_valid) {
                drive[i].addr = el_stat.bus_address;
                if (! el_stat.lu_valid) {
                    drive[i].lun = 0;
                } else {
                    drive[i].lun = el_stat.lun;
                }
            }
        }
    }
}
}
```

WARNINGS

Do not use LVM to configure multiple autochanger surfaces as one large file system. LVM was designed for on-line volumes. In an autochanger, only the disks actually in the drives are on-line, while the disks stored in their slots are off-line. If LVM is not carefully configured, thrashing of the autochanger disks result in undesirable I/O performance. Plan carefully for best performance.

Some non-HP media changer devices do not support the `SIOC_INIT_ELEM_STAT` and `SIOC_ELEMENT_STATUS` ioctls.

Some older media changer devices do not support the `SIOC_EXCHANGE_MEDIUM` ioctl. For these devices, multiple `SIOC_MOVE_MEDIUM` ioctl operations may be used to accomplish the same results, provided a suitable temporary element address may be found.

SEE ALSO

`insf(1M)`, `mknod(1M)`, `scsictl(1M)`, `ioctl(2)`, `scsi(7)`, `scsi_ctl(7)`.

NAME

blmode - terminal block mode interface

DESCRIPTION

This terminal interface adds functionality to the current *termio(7)* functionality to allow for efficient emulation of MPE terminal driver functionality. Most importantly, it adds the necessary functionality to support block mode transfers with HP terminals. The block mode interface only affects input processing and does not affect write requests. Write requests are always processed as described in *termio(7)*. In character mode the terminal sends each character to the system as it is typed. However, in block mode data is buffered and possibly edited locally in the terminal memory as it is typed, then sent as a block of data when the **Enter** key is pressed on the terminal. During block mode data transmissions, the incoming data is not echoed and no special character processing is performed, other than recognizing a data block terminator character. For subsequent character mode transmissions, the existing *termio* state continues to determine echo and character processing.

There are two parts of the block mode protocol. The first part is the block mode handshake, which works as follows:

- At the beginning of a read, a *trigger* character is sent to the terminal to notify it that the system is requesting a block of data. (The *trigger* character, if defined, is sent at the beginning of all reads, whether character or block. The *trigger* character must be defined for block mode reads.)
- After receiving the *trigger* character, the terminal waits until the user has typed data into the terminal's memory and pressed the terminal **Enter** key. The terminal then sends an *alert* character to the system to notify it that the terminal has a block of data to send.
- The system may then send user-definable cursor positioning or other data sequences to the terminal. When that is done, the system sends another *trigger* character to the terminal, repeating the cycle.

The second part of the block mode protocol is the block mode transmission. During this transmission of data, the incoming data is not echoed and no special character processing is performed, other than recognizing the data block termination character. It is possible to bypass the block mode handshake and have the block mode transmission occur after the first *trigger* character is sent.

To prevent data loss, XON/XOFF flow control should be used between the system and the terminal. The IXOFF bit should be set and the terminal strapped appropriately. If flow control is not used, it is possible for incoming data to overflow and be lost. (Note: some older terminals do not deal correctly with this flow control.)

It is possible to intermix both character mode and block mode data transmissions. If block mode transmissions are enabled, all transfers are handled as block mode transfers. When block mode transmissions are not enabled, character mode transmissions are processed as described in *termio(7)*. If block mode transmissions are not enabled, but an *alert* character is received anywhere in the input data, the transmission mode is switched to block mode automatically for a single transmission.

Read requests that receive data from block mode transmissions will not be returned until the transmission is complete; i.e., the terminal has transmitted all characters. If the read is satisfied by byte count or if a data transmission error occurs, any subsequent data will be discarded. The read waits until completion of the data transmission before returning.

The data block terminator character is included in the data returned to the user, and is included in the byte count. If the number of bytes transferred by the terminal in a block mode transfer exceeds the number of bytes requested by the user, the read returns the requested number of bytes, and the remaining bytes are discarded. The user can determine if data was discarded by checking the last character of the returned data. If the last character is not the terminator character, more data was received than was requested, and data was discarded.

If desired, the application program can provide its own handshake mechanism in response to the *alert* character by selecting the *OWNTERM* mode. With this mode selected, the driver completes a read request when the *alert* character is received. The second *trigger* is sent by the driver when the application issues the next read.

Several special characters (both input and output) are used with block mode. These characters and the normal values used for block mode are described below. The initial value for these characters is 0377, which causes them to be disabled.

CBTRIG1C	(DC1) is the initial <i>trigger</i> character sent to the terminal at the beginning of a read request.
CBTRIG2C	(DC1) is the secondary <i>trigger</i> character sent to the terminal after the <i>alert</i> character has been received.
CBALERTC	(DC2) is the <i>alert</i> character sent by the terminal in response to the first <i>trigger</i> character. It signifies that the terminal is ready to send the data block. The <i>alert</i> character can be escaped by preceding it with a backslash (\).
CBTERMC	(RS) is sent by the terminal after the block mode transfer is complete. It signifies the end of the data block to the computer.

The two *ioctl(2)* requests that apply to block mode use the **blmodeio** structure, which defined in **<blmodeio.h>**, and includes the following members:

```

unsigned long  cb_flags;      /* Modes */
unsigned char  cb_trig1c;    /* First trigger */
unsigned char  cb_trig2c;    /* Second trigger */
unsigned char  cb_alertc;    /* Alert character */
unsigned char  cb_termc;     /* Terminating char */
unsigned char  cb_replen;    /* cb_reply length */
char          cb_reply[];    /* optional reply */

```

The *cb_flags* member controls the basic block mode protocol:

```

CB_BMTRANS    0000001    Enable mandatory block mode transmission.
CB_OWNTERM    0000002    Enable user control of handshake.

```

The **CB_BMTRANS** bit is only effective when the **ICANON** flag in *termio(7)* is set. If **ICANON** is clear, all transfers are done in raw mode, regardless of the **CB_BMTRANS** bit. If **CB_BMTRANS** is not set, input processing is performed as described in *termio(7)*. During this time, if the *alert* character is defined and is detected anywhere in the input stream, the input buffer is flushed and block-mode handshake is invoked. The system then sends the *cb_trig2c* character to the terminal, and a block mode transfer follows. The *alert* character can be escaped by preceding it with a backslash (\).

If **CB_BMTRANS** is set, then all transmissions are processed as block mode transmissions. Block mode handshake is not required and data read is processed as block mode transfer data. Block mode handshake can still be invoked by receipt of an *alert* character as the first character received. Reads issued while the **CB_BMTRANS** bit is set cause any existing input buffer data to be flushed.

If **CB_OWNTERM** is set, reads are terminated upon receipt of a non-escaped *alert* character. No input buffer flushing is performed and the *alert* character is returned in the data read. This allows application code to perform its own block-mode handshaking. If the bit is clear, an *alert* character causes normal block mode handshaking to be used.

The initial **cb_flags** value is all-bits-cleared.

The **cb_trig1c** character is the initial *trigger* character sent to the terminal at the beginning of a read request. The initial value is undefined (0377); i.e., no *trigger* character is sent.

The **cb_trig2c** character is the secondary *trigger* character sent to the terminal after the *alert* character has been received. The initial value is undefined (0377).

The **cb_alertc** character is the *alert* character sent by the terminal in response to the first *trigger* character sent by the computer. It signifies that the terminal is ready to transmit data. The initial value is undefined (0377).

The **cb_termc** character is sent by the terminal after the block mode transfer has completed. It signifies the end of the data block to the computer. The initial value is undefined (0377).

The **cb_replen** member specifies the length in bytes of the **cb_reply** array. The maximum length of the **cb_reply** array is **NBREPLY** bytes. If set to zero, the *cb_reply* string is not used. It is initially set to zero.

The *cb_reply* array contains a string to be sent out after receipt of the *alert* character but before the second *trigger* character is sent by the computer. Any character can be included in the reply string. The number of characters sent is specified by **cb_replen**. The maximum length of the **cb_reply** array is **NBREPLY** bytes. The initial value of all characters in the **cb_reply** array is null.

On systems that support process group control, *ioctl* requests are restricted from use by background processes, unless otherwise noted for a specific request. An attempt to issue an *ioctl* request from a background process causes the process to block and may cause a SIGTTOU signal to be sent to the process group.

The primary *ioctl*(2) calls have the form:

```
int ioctl(int fildes, int request, struct blmodeio *arg);
```

Requests using this form include:

- | | |
|--------|---|
| CBGETA | Get the parameters associated with the block mode interface and store them in the <i>blmodeio</i> structure referenced by <i>arg</i> . This request is allowed from a background process. However, the information may be subsequently changed by a foreground process. |
| CBSETA | Set the parameters associated with the block mode interface from the <i>blmodeio</i> structure referenced by <i>arg</i> . The change is immediate. |

RETURN VALUE

Refer to *read*(2), *write*(2), and *ioctl*(2).

ERRORS

If an error value is returned during a read, it is possible for the user's buffer to be altered. In this case, the data in the user's buffer should be ignored because it is incomplete.

The global variable *errno* will be set to indicate the following error, in addition to those errors described on *read*(2), *write*(2), and *ioctl*(2):

- | | |
|-------|---|
| [EIO] | A read error occurred during the transmission of the block mode data block. |
|-------|---|

WARNINGS

The EIO error that is returned for read errors can be caused by many events. The read returns EIO for transmission, framing, parity, break, and overrun errors, or if the internal timer expires. The internal timer starts when the second *trigger* character is sent by the computer, and ends when the terminating character is received by the computer. The length of this timer is determined by the number of bytes requested in the read and the current baud rate, plus an additional ten seconds.

AUTHOR

The *blmode* driver was developed by HP.

SEE ALSO

termio(7).

NAME

cent - Centronics-compatible interface

DESCRIPTION

cent is a simple, widely used communication protocol most commonly associated with printers, plotters and scanners. It is an eight-bit parallel data interface with additional control signals from the host computer, and status signals from the peripheral.

The **cent** interface driver does no character processing; that is, it does not interpret the data being transferred between computer and peripheral. Therefore, all bytes sent to or received from a device are handled without alteration. The **cent** interface driver always operates in **raw mode**; therefore, any desired data interpretation must be performed by a user program (such as the "lp" spooler in conjunction with an appropriate model file). The **cent** driver supports six different handshake modes for data transfer. The last four bits of the minor number of the device special file specify the mode used. The format of the device minor number is:

0xII000A

where each letter after the "0x" prefix represents a single hexadecimal digit, as follows:

II Specifies the instance number of the centronic interface.

000 Always zero.

A Specifies the handshake mode. The handshake modes are:

- mode 1 Automatic handshaking using both ACK and BUSY.
Minor number format: **0xII0001**.
- mode 2 Automatic handshaking using only BUSY.
Minor number format: **0xII0002**.
- mode 3 Bidirectional read/write used for ScanJet.
Minor number format: **0xII0003**.
- mode 4 Stream mode. Data is essentially transmitted to the peripheral without any handshaking protocol.
Minor number format: **0xII0004**.
- mode 5 Pulsed mode using both ACK and BUSY for automatic handshaking. Similar to mode 1 except that the data strobe line, nSTROBE is pulsed for a fixed amount of time by the sender, then released.
Minor number format: **0xII0005**.
- mode 6 Pulsed mode, using only BUSY for automatic handshaking. Similar to mode 1 except that the data strobe line, nSTROBE is pulsed for a fixed amount of time by the sender, then released.
Minor number format: **0xII0005**.

Modes 1 and 2 support most HP *Jet series printers (LaserJet, DeskJet, QuietJet, etc.).

AUTHOR

cent was developed by HP.

SEE ALSO

lp(1), ioctl(2), intro(7), lp(7).

NAME

clone - opens a major and minor device pair on a STREAMS driver

DESCRIPTION

The **clone** driver is a "pass through" device driver that allows other drivers to select unique minor device numbers on each **open()**. In effect, the driver passes an open operation through to the other driver. This mechanism allows for multiple instantiations of a driver, each with a different minor number, through a single device file.

When the **clone** driver is opened, it is passed a major and minor device number by the operating system. The major number is the **clone** driver's major number (72), and the minor number is the major number of the driver the user wishes to clone (referred to here as the target driver). The **clone** driver calls the open routine of the target driver with the **CLONEOPEN** flag which specifies a clone open. The target driver's open routine allocates an unused minor number. The target driver must use **makedev** to make a new device number for the newly created device, and must set ***devp** to the new device number returned by **makedev**. The new device number is returned to the **clone** open through ***devp**. The **clone** open then returns to the user a file descriptor that points to the new instantiation of the target driver.

The **echo** driver is an example of a clonable driver.

Notes

It is not possible to do multiple opens of a device with the same major and minor number using the **clone** driver. This is because the **clone** driver is only given the major number of the driver to be cloned, and that driver will then select a minor number which has not been opened.

When called with a pathname which corresponds to the clonable driver, **stat()** will return different results than **fstat()** when it is called on a file descriptor returned from **open()** of the same clonable driver pathname.

RETURN VALUES

If the **clone** driver is given an invalid minor number, or if the driver indicated is not a clonable driver, the **open()** fails and **errno** is set to [ENXIO].

SEE ALSO

open(2), **fstat(2)**.

NAME

console, systty, syscon - system console interface

DESCRIPTION

/dev/console provides a *termio* interface to the device configured as the system console. The *init*(1M) man page discusses the uses of **/dev/systty** and **/dev/syscon**.

Output data normally sent to the console, either through **/dev/console** or generated by a kernel **printf**, may be redirected to another terminal or pseudo-terminal device through the **TIOCCONS ioctl()**. See *termio*(7) for details.

FILES

/dev/console
/dev/systty
/dev/syscon

SEE ALSO

termio(7), *init*(1M).

STANDARDS CONFORMANCE

console: SVID2, SVID3, XPG2

C

NAME

ct - Command-Set 80 (CS/80) cartridge tape access

DESCRIPTION

This entry describes the actions of the general HP-UX Command-Set 1980 (CS/80) cartridge tape drivers when referring to a CS/80 cartridge tape as either a block- or character-special (raw) device.

Cartridge tapes are designed to work optimally as "streaming" devices, and are not designed to start and stop frequently. Technically, they are "random access" devices such as disks, but such access is both less efficient and more stressful than streaming mode. Thus it is possible to use a cartridge tape as a file system, or in general access it randomly, but such use will more rapidly wear either or both the tape drive and the media.

Cartridge tape units in either CS/80 disk drives or in stand-alone devices can be accessed as blocked or raw devices.

Block special files access cartridge tapes via the system's normal buffering mechanism. Buffering is done in such a way that concurrent access through multiple opens or a mount of the same physical device do not get out of phase. Block special files may be read and written without regard to physical cartridge tape records. Each I/O operation results in one or more logical block transactions. In general, this mode is not recommended as it stresses the hardware.

There is also a **raw** interface via a character special file which provides for direct transmission between the cartridge tape and the user's read or write buffer. A single read or write operation results in exactly one transaction. Therefore raw I/O is considerably more efficient when many bytes are transmitted in a single operation because blocked cartridge tape access requires potentially several transactions and does not transmit directly to user space.

In raw I/O, there may be implementation dependent restrictions on the alignment of the user buffer in memory and its maximum size. Also, each transfer must occur on a record boundary, and must read a whole number of records. The record size is a hardware-dependent value.

Selecting the proper buffer size when accessing a cartridge tape device through the raw interface is critical to the performance of the cartridge tape device and other devices connected on the same HPIB. A large buffer in certain situations can increase performance but has the potential to block other devices on the HPIB until all the data for a request has been transferred. On the other hand when a small buffer is used and the application is unable to keep the cartridge tape device streaming, performance and the wear and tear of the device suffer because of tape repositioning. The optimal solution is to keep the tape streaming while using a small buffer. To select the proper buffer size, consider two factors: the cartridge tape device being accessed and the application which is accessing the cartridge tape device.

Some cartridge tape units (see *DEPENDENCIES*) support a feature called immediate report mode. During writing, this mode enables the drive to complete a write transaction with the host before the data has actually been written to the tape from the drive's buffer. This allows the host to start gathering data for the next write request while the data for the previous request is still in the process of being written. During reading, this mode enables the drive to read ahead after completing a host read request. This allows the drive to gather data for future read requests while the host is still processing data from the previous read request. When data is requested or supplied at a sufficient rate, immediate report mode allows the drive to stream the tape continuously across multiple read/write requests, as opposed to having to reposition the tape between each read/write request. Repositioning adds to the wear and tear of the cartridge tape device and decreases the performance. Some cartridge tape devices (see *DEPENDENCIES*) do not support immediate report mode and as such cannot stream across multiple requests.

If the cartridge tape device being accessed supports immediate report mode and the application can maintain a data rate that allows the cartridge tape device to stream multiple requests, a small buffer (1 Kbyte to 12 Kbytes) is suggested so that the HP-IB is not blocked for a significant amount of time. For cartridge tape devices that do not support immediate report mode or applications that cannot maintain a data rate that allows the cartridge tape device to stream multiple requests, a large buffer (64 Kbytes) is suggested so that the number of tape repositions is reduced.

Each raw access is independent of other raw accesses and of block accesses to the same physical device. Thus, transfers are not guaranteed to occur in any particular order. Having multiple programs access the cartridge tape is, in effect, random access, and is subject to the warnings above.

In raw I/O, each operation is completed to the device before the call returns. For block-mode writes, the data may be cached until it is convenient for the system to write it. In addition, block-mode reads potentially do a one (or more) block read-ahead. The interaction of block-mode and raw access to the same

cartridge tape is not specified, and in general is unpredictable. Because block-mode writes can be delayed, it is possible for a program to generate requests much more rapidly than the drive can actually process them. Flushing a large number of requests could take several minutes, and during that time the system will not have use of the buffers taken by these requests, and thus will suffer a possibly severe performance degradation. If the tape is integral with the system disk, very little disk activity may be possible until the buffers are flushed.

Cartridge tape device file names are in the following format:

```
/dev/[r]ct/[r]c#d#[s#]
```

where the first **r** indicates a raw interface to the cartridge tape, the second **r** is reserved to indicate that this cartridge tape is on a remote system, the **c#** indicates the controller number, the **d#** optionally indicates the drive, and the **s#** optionally indicates a section number. The assignment of controller, drive, and section numbers is described in the system administrator's manual for your system.

WARNINGS

Like disks, the cartridge tape units in CS/80 disk drives can be accessed as blocked or raw devices. However, using a cartridge tape as a file system severely limits the life expectancy of the tape drive. Tapes should be used only for system back-up and other needs where data must be stored on tape for transport or other purposes.

ct does not support access of DDS and QIC cartridge tape devices.

DEPENDENCIES

HP 7941CT/HP 9144A/HP 35401

These cartridge tape devices support the immediate report mode.

HP 7942/HP 7946

These cartridge tape devices support the immediate report mode. The use of a small buffer size is not recommended with these shared controller devices when there is simultaneous access to the disk, because the disk accesses will prevent proper tape streaming.

HP 7908/HP 7911/HP 7912/HP 7914

These cartridge tape devices do not support the immediate report mode.

AUTHOR

ct was developed by HP and AT&T.

SEE ALSO

mknod(1M), tcio(1), disk(7), intro(7), mt(7).

NAME

ddfa - Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software

DESCRIPTION

The Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software allows access from HP-UX system utilities and user applications to terminal servers using standard HP-UX structures. DDFA provides an interface to remote LAN-connected terminal server ports that is similar to the interface for local directly-connected ports.

The basic principle is that a daemon is created for each configured terminal server port based on information in a configuration file (a Dedicated Ports file). When the daemon is spawned, it takes a **pty** from the pool and creates a device file with the same major and minor number as the **pty** slave. The device file is known as the "pseudonym" and utilities and applications use the pseudonym to access the terminal server port by exercising standard HP-UX system functions (**open()**, **close()**, **read()**, **write()**, and **ioctl()**). The daemon listens on the **pty** until an application does an **open()** on the pseudonym. It then sets up and manages the connection to the terminal server port until the application does a **close()** on the pseudonym. The end result is that the terminal server port is addressed via a device file, but the mechanism that makes it happen is transparent to the user. A second configuration file (a port configuration file) contains information to profile the terminal server port.

DDFA consists of the following items:

dp	Dedicated Ports file. This text file contains the information that DDFA needs to set up and manage a connection between a pseudonym and a terminal server port. The dp file is parsed by the Dedicated Port Parser (dpp) which spawns an Outbound Connection Daemon (ocd) for each outbound connection specified in the file. The dp file is also used by the HP-UX Telnet daemon (telnetd) to identify incoming connections from a DTC and map them to a pseudonym (the Telnet port identification feature).
pcf	Port Configuration File. This text file is used by DDFA to profile the terminal server port. The generic name of the template file is pcf . A port configuration file is referenced by an entry in the Dedicated Ports file (dp).
dpp	Dedicated Port Parser. This command parses the Dedicated Ports file (dp) and spawns an Outbound Connection Daemon (ocd) for each valid entry in the dp file. It can be run from the shell or it can be included in a system initialization script to automatically run the DDFA software each time the system is booted.
ocd	Outbound Connection Daemon. This daemon manages the connection and data transfer to the remote terminal server port. Normally, it is spawned by the Dedicated Ports Parser (dpp), but it can be run directly from the shell. As it starts, it creates its pseudonym for the connection. As it terminates normally, it removes the pseudonym. If the pseudonym is removed while it is running, ocd will terminate with an error condition.
ocdebug	Outbound Connection Daemon debug mode. This is a special version of ocd that contains debugging code. It must be run from the shell.

CONFIGURATION

There are two basic steps to configuring the DDFA software:

- Enter information in the **dp** file.
- Enter information in the port configuration files.

Configuring the dp File

The **dp** file contains one line for each outbound connection that is to be established and one line for each incoming connection request. A default file `/usr/examples/ddfa/dp` should be copied to a new file and the copy edited as needed. It is recommended that a directory be created to hold the **dp** file and the port configuration files.

Each line of the **dp** file must contain the location of the terminal server port and the location of the pseudonym. In addition, for an outbound connection, the port configuration file must be specified and a logging level may be specified.

Configuring the Port Configuration Files

A port configuration file is used to configure individual terminal server ports. A master port configuration file is `/usr/examples/ddfa/pcf`. In practice, it is renamed for each port that needs different configuration values and the values are altered appropriately for the device attached to the port. It is recommended that a directory be created to hold the port configuration files and the `dp` file.

Each line of a port configuration file must consist of a name of a variable and its value. The variable-value pairs contain information on how to open a connection to a terminal server port, how to close a connection to a terminal server port, and how to manage the data transfer to a terminal server port.

Configuring a System Initialization Script

DDFA can be run at boot time by including a reference to `dpp` in a system initialization script. It is recommended that the `-k` option be used when running `dpp` in this environment.

KILLING DAEMONS

Note that `ocd` should be killed using `kill -15`. Do not use `kill -9` for this purpose as it does not remove the device file. `ocd` verifies the validity of an existing pseudonym before trying to use it. `dpp` and `ocd` use data stored in the file `/var/adm/utmp.dfa` to verify whether a process still owns a pseudonym before taking it over. If `ocd` finds an unowned pseudonym, it uses it.

ERROR HANDLING

When `ocd` receives a serious error condition, such as when the LAN goes down, it transmits the error condition to the application by closing the `pty`. Any `open()`, `close()`, `read()`, or `write()` to the pseudonym returns the error condition `0 bytes read`. If the pseudonym is the controlling terminal for the group to which the application belongs, `SIGHUP` is sent to all the processes in the group, including the application.

ioctl() LIMITATIONS

Not all `ioctl()` functionality is available, due to the lack of a protocol that allows the transmission of such commands over the LAN to the remote port.

termio Attribute Limitations

The main restrictions on `termio` attributes (see *termio(7)*) include modem signal control and parity checking. The following are not available:

CBAUD IGNPAR INPCK IXANY IXOFF PARMRK

ioctl() Request Limitations

The following `ioctl()` request limitations apply:

CSTOPB flag	DTC only supports one stop bit.
CSIZE	DTC only supports 8 bits per character. Value cannot be modified.
PARODD flag	DTC offers static configuration to handle even or odd parity. It also handles auto parity detection for even or odd parity.
PARENB flags	Enabling/disabling done via static configuration. No programmatic interface supplied.
INPCK flag	No way to separate input from output parity features.
IGNPAR flag	Cannot be configured on DTC.
PARMRK	Bad characters are forwarded to the system without marking them with OFFH or OH.
CBAUD	Speed is part of static configuration.
IXOFF flag	Flow control is enabled if the DTC static configuration specifies an ASCII access mode. If binary is selected, no flow control is provided.
IXON flags	Pacing of output to a terminal via a programmatic interface is enabled when ASCII mode is selected in static port configuration and disabled when binary mode is selected.
IXANY flag	DTC does not offer the ability to restart output on any character received if XOFF was previously received.

HUPCL flag	DDFA does not support the hanging up of modem signals on the last close of the device file. If the modem signals used on the DTC drop, the connection is closed.
CLOCAL flag	Not supported.
c_flags	IENQACK not supported. OFILL , OFDEL , NLDLY , CRDLY , TABDLY , BSDLY , FFDLY not supported by Telnet port identification software.
BINARY mode flags	Part of static configuration is done in DTC Manager by selecting binary mode. If switching is enabled, binary can be selected at user interface level. There is no way to automatically negotiate binary mode when proper termio flags are reset when using telnetd . Binary/ASCII switching is possible with DDFA. The DTC cannot support large reads in pure binary mode, so transferred blocks of data should not be more than 256 bytes. If half-duplex with remote acknowledgement is implemented, binary applications can be supported.

ioctl() System Call Requests

The following **ioctl()** system call limitations apply:

TCSEBK	The ability to send a break without waiting for previous data to be sent is not provided at the system level in telnetd or DDFA. Receiving a Telnet break command in the DTC allows it to generate a break on asynchronous ports.
TCFLSH	The DTC output queue cannot be flushed.
Hardware handshake request	Not supported on DTC.
TCXONC	Local handshake cannot be disabled on DTC.
MCGETA	Not supported.
MCSETA , MCSETAF , MCSETAW	There is no way to separately set modem lines of a DTC port.
MCGETT	Modem timers, CD timer, connect timer, and disconnect cannot be configured.
CCITT simple, and direct call-in/call-out modes	DTC cannot handle simple mode because there is programmatic interface for modem signals. Call-in mode cannot be simulated if the port is opened, because modem signals (or the call) must be present within 2 minutes or the connection is cleared.
DACIDY get device adapter info	No way to get device adapter information.
Download ioctl() DACRADDR , DACDLADDR , DACDLGO , DACDLVER	No programmatic call to download the DTC.
DACHWSTATUS , DACSELFTEST , DACLOADED , DACISBROKE status	No programmatic interface to get such info.
DACLOOPBACK DACSUBTEST port test	

WARNINGS

In order to ensure that commands (such as *ps*) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&l 2>/dev/null
```

The printer interface scripts reside in the directory `/etc/lp/interface`. The line must be added just prior to the final `exit` command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

```
/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/utmp.dfa
```

SEE ALSO

dpp(1M), ocd(1M), ocdebug(1M), ioctl(2), dp(4), pcf(4), ioctl(5), termio(7).

NAME

diag0 - diagnostic interface to I/O subsystem

DESCRIPTION

diag0 is the diagnostic pseudo-driver used by the on-line diagnostic subsystem. The kernel sends diagnostic events here for logging by the diagnostic subsystem. The diagnostic subsystem also sends diagnostic requests to the kernel via **diag0**.

WARNINGS

If the diagnostic subsystem does not read diagnostic events from **diag0** as fast as they are logged by the kernel, a warning message **Warning: DIAG0 -- message queue full** appears on the console.

AUTHOR

diag0 was developed by HP.

FILES

/stand/vmunix
/dev/diag0
/dev/diag/diag0
/dev/diag directory containing diagnostic device files

SEE ALSO

sysdiag(1M), diaginit(1M).

Online Diagnostic Subsystem manuals.

NAME

disk - direct disk access

DESCRIPTION

This entry describes the actions of HP-UX disk drivers when referring to a disk as either a block-special or character-special (raw) device.

Device File Naming Conventions

Standard disk device files are named according to the following conventions:

Block-mode Devices	<code>/dev/dsk/cxt_yd_n[sm]</code>
Character-mode Devices	<code>/dev/rdsk/cxt_yd_n[sm]</code>

where component parts of the filename are constructed as follows:

- c** Required. Identifies the following hexadecimal digits as the “Instance” of the interface card.
- x** Hexadecimal number identifying controlling bus interface, also known as the “Instance” of this interface card. The instance value is displayed in the *ioscan*(1M) output, column “I” for the H/W Type, “INTERFACE”.
Required.
- t** Identifies the following hexadecimal digits as a “drive number” or “target”.
Required.
- y** Hexadecimal number identifying the drive or target number (bus address).
Required.
- d** Identifies the following hexadecimal digits as a “unit number”.
Required.
- n** Hexadecimal unit number within the device.
Required.
- s** Optional. Defaults to that corresponding to whole disk. Identifies the following value as a “section number”.
- m** Required if **s** is specified. Defaults to section 0 (zero), whole disk. Drive section number.

Assignment of controller, drive, logical unit and section numbers is described in the system administrator manuals for your system.

Block-special access

Block-special device files access disks via the system's block buffer cache mechanism. Buffering is done in such a way that concurrent access through multiple opens and mounting the same physical device is correctly handled to avoid operation sequencing errors. The block buffer cache permits the system to do physical I/O operations when convenient. This means that physical write operations may occur substantially later in time than their corresponding logical write requests. This also means that physical read operations may occur substantially earlier in time than their corresponding logical read requests.

Block-special files can be read and written without regard to physical disk records. Block-special file `read()` and `write()` calls requiring disk access result in one or more `BLKDEV_IOSIZE` byte (typically 2048 byte) transfers between the disk and the block buffer cache. Applications using the block-special device should ensure that they do not read or write past the end of last `BLKDEV_IOSIZE` sized block in the device file. Because the interface is buffered, accesses past this point behave unpredictably.

Character-special access

Character-special device files access disks without buffering and support the direct transmission of data between the disk and the user's read or write buffer. Disk access through the character special file interface causes all physical I/O operations to be completed before control returns from the call. A single read or write operation up to `MAXPHYS` bytes (typically 64 Kbytes or 256 Kbytes) results in exactly one disk operation. Requests larger than this are broken up automatically by the operating system. Since large I/O operations via character-special files avoid block buffer cache handling and result in fewer disk operations, they are typically more efficient than similar block-special file operations.

There may be implementation-dependent restrictions on the alignment of the user buffer in memory for character special file `read()` and `write()` calls. Also, each read and write operation must begin and end on a sector boundary and must be a whole number of sectors in size. The sector size is a hardware-

dependent value that can be queried with the **DIIOC_DESCRIBE** ioctl call, which is described below.

In addition to reading and writing data, the character-special file interface can be used to obtain device specific information and to perform special operations. These operations are controlled through use of ioctl calls. Details related to these ioctls are contained in `<sys/diskio.h>`.

The **DIIOC_DESCRIBE** ioctl can be used to obtain device specific identification information. The information returned includes the disk's model identification, the disk interface type, and the disk's sector size.

The **DIIOC_CAPACITY** ioctl can be used to obtain the capacity of a disk device in **DEV_BSIZE** units. (**DEV_BSIZE** is defined in `<sys/param.h>`).

The **DIIOC_EXCLUSIVE** ioctl can be used to obtain and release exclusive access to a disk device. Exclusive access is required for some special operations, such as media reformatting, and may be desirable in other circumstances. The value one specifies that exclusive access is requested. The value zero specifies the exclusive access should be released. Exclusive access causes other open requests to fail. Exclusive access can only be granted when the device is not currently opened in block-mode and there is only one open file table entry for that disk device (the one accessible to the exclusive access requester).

ERRORS

The following errors can be returned by a disk device driver call:

[EACCES]	Required permission is denied for the device or operation.
[ENXIO]	If resulting from an open() call, this indicates there is no device at the specified address. For other calls, this indicates the specified address is out of range or the device can no longer be accessed.
[EINVAL]	From an open() call: the device is not a disk device. For other calls: Invalid request or parameter.
[EIO]	I/O error (e.g., media defect or device communication problem).

WARNINGS

The interaction of block-special and character-special file access to the same **BLKDEV_IOSIZE**-sized block is not specified, and in general is unpredictable.

On some systems, having both a mounted file system and a block special file open on the same device can cause unpredictable results; this should be avoided if possible. This is because it may be possible for some files to have private buffers in some systems.

Although disk devices have historically had small (typically 512-byte) block sizes, some disk devices (such as optical disks and disk arrays) have relatively large block sizes. Applications using direct raw disk access should use **ioctl()** calls to determine appropriate I/O operation sizes and alignments.

Any disk with removable media (for example, floppy or CD-ROM) containing a mounted file system should not be removed prior to being unmounted. Removal of disk media containing mounted file systems is likely to result in file system errors and system panics.

DEPENDENCIES

disc1/disc2/disc3/disc4

Devices whose sector size is less than **DEV_BSIZE** must be accessed on **DEV_BSIZE** boundaries and with transfer sizes that are multiples of **DEV_BSIZE**. Disk "sections" 0 (zero) and 2 (two) have exchanged meanings with HP-UX Release 10.0 and beyond. Whole disk is section 0.

AUTHOR

disk was developed by HP and AT&T.

SEE ALSO

mkknod(1M), **ct(7)**, **intro(7)**, **ioscan(1M)**.

System Administrator manuals included with your system.

NAME

fddi - Fiber Distributed Data Interface Tools

DESCRIPTION

This manual entry provides general information on the HP-UX FDDI (Fiber Distributed Data Interface) link tools.

Tools for HSC and EISA FDDI

fddiif Display FDDI driver operating statistics in real-time.

fddilink Display FDDI interface and network characteristics.

Tools for PCI FDDI

fddipciadmin Display FDDI interface, driver and network statistics and characteristics.

Tools for HP-PB FDDI and S700 Built-in FDDI

fddiinit Initialize FDDI network interface; connect to FDDI network.

fddinet Display logical FDDI ring map information.

fddistat Display status information for the FDDI interface.

fddistop Stop and reset the FDDI interface.

General Link Tools

ifconfig Configure or display IP interface information.

ioscan Scan system for devices, display hardware path, class and description.

lanadmin Display link information, RFC 1213 MIB II statistics, reset the interface card, perform card self-test.

lanscan Scan system for LAN devices, display hardware path, station address (canonical format), instance number, hardware state, interface name, IP interface state, Network Management ID.

linkloop Verify LAN connectivity with a link-level test frame. Requires the Physical Point of Attachment number (PPA) of the local interface and station address of the remote station in canonical format. The PPA is the numeric portion of the interface name. For example, the interface lan2 has the PPA 2. Use **lanscan** on the remote system to get the remote station address in canonical format.

netfmt Format network trace and log files created with the nettl utility (see below). FDDI entity names are FDDI (HP-PB and S700 built-in FDDI), EISA_FDDI, HSC_FDDI and PCI_FDDI.

netstat Show network statistics for IP interfaces and upper-layer protocols.

nettl Start/stop network tracing and logging. FDDI entity names are FDDI (HP-PB and S700 built-in FDDI), EISA_FDDI, HSC_FDDI and PCI_FDDI.

ping Verify IP/ICMP connectivity to remote system.

syslogd Read and log system messages (the FDDI drivers log system messages using syslog routines).

what Get version identification information. Use **what /stand/vmunix** to determine if the correct FDDI driver has been installed, according to the following list:

fddi (for HP-PB FDDI), fddi0 (for EISA FDDI), fddi2 (for S700 built-in FDDI), fddi3 (for HSC FDDI), fddi4 (for PCI FDDI).

FDDI station addresses are represented as a series of hexadecimal digits using canonical format, or wire or "native" format. The canonical format uses Least Significant Bit (LSB) order. Wire or "native" format uses Most Significant Bit (MSB) order. Most HP-UX FDDI and network utilities use canonical format (**fddi-link** and **fddiif** are two exceptions). The **fddinet** and **fddistat** commands display the addresses in canonical format by default and have a **-n** option to display the addresses in "native" (wire) format.

SEE ALSO

fdiinit(1M), fddinet(1M), fddipciadmin(1M), fddistat(1M), fddistop(1M), ioscan(1M), lanadmin(1M), lanscan(1M), netfmt(1M), netstat(1), nettl(1M), ping(1M), syslogd(1M), what(1).

NAME

floppy - direct flexible or “floppy” disk access

DESCRIPTION

Flexible disk devices are removable-media disk devices that are typically used to share data with other systems. Media types are identified by physical size (such as 3.5-inch and 5.25-inch), number of data surfaces (or sides), and data density. By convention, flexible disk devices are named using the same conventions as those used for other disk devices (see *disk(7)*).

Data can be stored on flexible disk media in a variety of logical formats. The capacity of these devices is generally too small to hold useful HP-UX file systems. Instead, DOS or LIF file systems (see *dosif(4)* and *lif(4)* for a detailed description of these file systems) are commonly used. Data can also be stored in an archive-utility format. For example, **tar** and **cpio** are commonly used to share data with other HP-UX systems (see *tar(1)* and *cpio(1)*).

Logical volumes are not supported on flexible disks. The floppy disk does not support partitions. Section zero (0) is the only accessible partition.

In addition to the various logical formats, data can be stored on flexible disk media in a variety of physical data formats called geometries. The following parameters are used to describe a flexible disk geometry:

<i>heads</i>	Number of surfaces (or sides) on the media that contain valid data.
<i>tracks</i>	Number of tracks on a single media surface or side (the term cylinders is sometimes also used). This value does not include spare tracks.
<i>sectors</i>	Number of sectors in a single track. The number of sectors that can fit on a track depends on the bit density (as controlled by transfer rate and media rotation rate) and the sector size.
<i>sector size</i>	Number of bytes in a logical sector. Since all I/O operations must be an integral number of sectors in length, this parameter also indicates the minimum character-special file I/O size.
<i>transfer rate</i>	Media data rate in Kbits per second. The transfer rate is an indirect means of representing bit density. Bit density is measured in bits per radian, and is the formal intra-track data density parameter for standard specification. Transfer rate is generally used to program flexible media devices and is therefore more appropriate for this interface. Since the media rotation rate for most flexible disk devices is standard, conversion between these two representations is straight-forward.
<i>track density</i>	Number of tracks per inch. Some low density formats can be supported on high-density drives by skipping tracks during head stepping.
<i>data encoding</i>	Encoding method used to store data. FM (frequency modulation) and MFM (modified frequency modulation) are the most common encoding methods.

The following table shows some useful flexible disk media geometries (without density information). The right-most column indicates which **mediainit -f** option should be used to format media to the indicated geometry (see *mediainit(1M)*).

Media Type	Use	Capacity	Heads	Tracks	Sectors	Sector Size	-f
3.5in DS DD	DOS 720K	630,784	2	77	16	256	1
3.5in DS DD		655,360	2	80	16	256	21
3.5in DS DD		655,360	2	80	8	512	26
3.5in DS DD		709,632	2	77	9	512	2
3.5in DS DD		737,280	2	80	9	512	16
3.5in DS DD		788,480	2	77	5	1024	3
3.5in DS HD'	DOS 1.2M	1,228,800	2	80	15	512	26
3.5in DS HD'		1,261,568	2	77	8	1024	22
3.5in DS HD	DOS 1.44M	1,261,568	2	77	32	256	1
3.5in DS HD		1,419,264	2	77	18	512	2
3.5in DS HD		1,474,560	2	80	18	512	16
3.5in DS HD		1,567,960	2	77	10	1024	3
3.5in DS HD		1,638,400	2	80	10	1024	23
5.25in DS	DOS 360K	368,640	2	40	9	512	2
5.25in DS DD	DOS 1.2M	655,360	2	80	8	512	26
5.25in DS HD		1,228,800	2	80	15	512	16
5.25in DS HD		1,261,568	2	77	8	1024	24

The following table shows the density information for some standard flexible disk media.

MediaType	BitDensity	RPM	TransferRate	TrackDensity	DataEncoding
3.5in DS DD	7,958	300	250	135	MFM
3.5in DS HD'	13,262	360	500	135	MFM
3.5in DS HD	15,916	300	500	135	MFM
5.25in DS	7,958	300	250	48	MFM
5.25in DS DD	7,958	300	250	96	MFM
5.25in DS HD	13,262	360	500	96	MFM

Abbreviations are interpreted as follows:

DS Double-sided media
 DD Double-density media
 HD High-density media
 HD' High-density media (alternate media rotation rate)

Normally each **open()** call causes the device and/or floppy device driver to attempt to determine the geometry of the installed media. As a result, the current flexible disk geometry is set to the supported geometry that matches the physical data format on the media currently installed in the device. To maintain reasonable open times, not all possible media geometries are checked. Therefore, it is possible that a flexible disk medium may contain valid data even though its format is not recognized. This automatic geometry sensing functionality may be disabled in some drivers by use of the **O_NDELAY** flag in the **open()** call or device driver dependent minor numbers. The Series 800 does not support this flag.

The **FLOPPY_GET_INFO** ioctl indicates the characteristics and current status of a floppy device. Information for interpreting the **media** and **data_encoding** fields can be found in **<sys/flopdy.h>**. The following macros are defined for decoding the **status** and **valid** fields. These macros return a non-zero (true) value for the associated conditions.

```

FLOPPY_NO_MEDIA(x)           /* Drive is empty */
FLOPPY_BLANK_MEDIA(x)        /* Media geometry is not recognizable */
FLOPPY_WRITE_PROT(x)         /* Media is write protected */
FLOPPY_MEDIA_CHANGED(x)      /* Media has changed since last status */
FLOPPY_HIGH_DENSITY(x)       /* Media has high density indication */

```

The argument **x** in the above macros refers to the **status** or **valid** fields of the **FLOPPY_GET_INFO** ioctl. Some floppy devices or floppy device drivers may be unable to determine some status information. The **valid** field indicates whether or not the corresponding status information is meaningful. Applying a macro to the **valid** field indicates whether or not the application of that same macro to the **status** field results in a valid device status.

The **FLOPPY_GET_GEOMETRY** ioctl can be used to determine the flexible disk device's current media geometry. Current geometry parameters are indicated in the fields of the resultant **floppy_geometry** structure.

The **FLOPPY_SET_GEOMETRY** ioctl can be used to specify the desired media geometry. Exclusive access to the device, obtained through use of the **DIOC_EXCLUSIVE** ioctl (see *disk(7)*), is required prior to setting the media geometry. Exclusive access is necessary to ensure that other applications are not affected. The Series 800 does not support this ioctl.

The **FLOPPY_FORMAT_TRACK** ioctl can be used to reformat a media track. Exclusive access to the device, obtained through use of the **DIOC_EXCLUSIVE** ioctl (see *disk(7)*), is required prior to reformatting to ensure that other applications are not affected. The Series 800 does not support this ioctl.

Flexible disk devices support the generic disk ioctls (see *disk(7)*), typically used for hard disk devices. Flexible disk device drivers may also support driver specific ioctls (see the appropriate driver manual section).

The header file `<sys/floppy.h>` has useful information for flexible-media device control. The following is included from `<sys/floppy.h>`:

```
/* ioctls for flexible (floppy) disk devices */
#define FLOPPY_GET_INFO      _IOR('F', 1, struct floppy_info)
#define FLOPPY_GET_GEOMETRY _IOR('F', 2, struct floppy_geom)
#define FLOPPY_SET_GEOMETRY _IOW('F', 3, struct floppy_geom)
#define FLOPPY_FORMAT_TRACK _IOW('F', 4, struct floppy_format)

/* structure for FLOPPY_GET_INFO ioctl */
struct floppy_info {
    unsigned    media;
    unsigned    status;
    unsigned    valid;
};

/* structure for FLOPPY_GET_GEOMETRY and FLOPPY_SET_GEOMETRY ioctls */
struct floppy_geometry {
    unsigned    heads;
    unsigned    tracks;
    unsigned    sectors;
    unsigned    sector_size;
    unsigned    transfer_rate;
    unsigned    track_density;
    unsigned    data_encoding;
};

/* structure for FLOPPY_FORMAT_TRACK ioctl */
struct floppy_format {
    unsigned    head;
    unsigned    track;
    unsigned    interleave;
};
```

ERRORS

The following errors can be returned by a flexible-disk device-driver call:

[EACCES]	Required permission is denied for the the device or operation.
[ENXIO]	If resulting from an open call, this indicates there is no device at the specified address. For other calls, this indicates the specified address is out of range or the device can no longer be accessed.
[EINVAL]	From an open() call: the device is not a floppy device. For other calls: Invalid request or parameter.
[EIO]	I/O error (e.g., media defect or device communication problem).

WARNING

Media removal and/or replacement while the device is open is not supported. A floppy disk containing a mounted file system should not be removed prior to being unmounted. Removal of floppy disks containing mounted file systems is likely to result in file system errors, and system panics.

Reformatting flexible disk media from one geometry to another that differs only in that it has fewer tracks can cause the automatic geometry sensing functionality of `open()` to fail to recognize the new media geometry. Bulk erasing (degaussing) the media or formatting the media to a substantially different geometry prior to reformatting prevents automatic geometry sensing problems.

Single track formatting may not be supported by some floppy devices.

Some devices permit configuration for geometries which they are unable to support. The `FLOPPY_SET_GEOMETRY` ioctl for such a configuration may not result in an error. However, subsequent I/O operations will fail.

DEPENDENCIES

Series 700

Series 700 device special files reside in the directories `/dev/rfloppy` and `/dev/floppy`.

Series 800

A flexible disk can be opened with no media. However, capacity changes resulting from subsequent media changes will not be recognized. (See `scsi_disk(7)`).

Series 800 device special files reside in the directories `/dev/rdisk` and `/dev/dsk`.

The following table of Series 800 Flexible Disk Media shows the disk geometries supported. The media inserted in the drive is formatted depending on the presence or absence of the HD hole in the flexible disk cartridge. HD media is formatted as a 1.44M diskette and DD media is formatted as a 720K diskette.

Media Type	Use	Capacity	Heads	Tracks	Sectors	Sector Size
3.5in DS DD	DOS 720K	737,280	2	80	9	512
3.5in DS HD	DOS 1.44M	1,474,560	2	80	18	512

The following table shows the density information for the flexible disk media supported:

MediaType	BitDensity	RPM	TransferRate	TrackDensity	DataEncoding
3.5in DS DD	7,958	300	250	135	MFM
3.5in DS HD	15,916	300	500	135	MFM

The `FLOPPY_GET_INFO` and `FLOPPY_GET_GEOMETRY` ioctls are supported on Series 800.

SEE ALSO

`disk(7)`, `mediainit(1)`, `mknod(1M)`, `dosif(4)`, `lif(4)`, `scsi(7)`, `scsi_disk(7)`.

NAME

framebuf - information for raster frame-buffer devices

SYNOPSIS

```
#include <sys/framebuf.h>
```

DESCRIPTION

Frame-buffer devices are raster-based displays. These devices use memory-mapped I/O to obtain much higher performance than possible with tty-based graphic terminals. Frame-buffer devices can be accessed directly using this interface, although access through the STARBASE libraries is recommended (see *starbase(3G)*). Direct access to frame-buffer devices entails precise knowledge of the frame-buffer architecture being used. Input cannot be piped into or redirected to frame-buffer devices because they are not serial devices.

Each frame-buffer device is associated with a character special file. Major and minor numbers for frame-buffer devices are implementation-dependent. The minor numbers for these devices denote different frame buffers. Implementation-specific details are discussed in the appropriate systems administrator's manuals.

Communication with a frame-buffer device begins with an *open(2)* system call. Multiple processes can have the frame-buffer device open concurrently.

close(2) invalidates the file descriptor associated with the frame-buffer device. After a *close* system call, any access to the frame-buffer device address range might result in a memory fault and the signal SIGSEGV being sent to the process (see *signal(2)*). A process cannot unmap the frame buffer from its address space after the frame-buffer special file is closed. To unmap a frame buffer, use the GCUNMAP *ioctl(2)* call (see below).

Once a process acquires a lock for the frame-buffer device, it must unlock it explicitly before calling *close(2)*; see GCUNLOCK below.

read(2) and *write(2)* system calls are undefined and always return an error. In this case **errno** is set to ENODEV.

The *ioctl(2)* system call is used to control a frame-buffer device. The *select(2)* system call is used to test the frame-buffer device for exceptional conditions. Interrupts from the graphic hardware are considered exceptional conditions. An exceptional condition is automatically cleared after any process that opens the frame-buffer device is notified of the exception by a *select(2)* call. A call to *select(2)* for read or write on the file descriptor associated with the frame-buffer device returns a false condition in the read and write bit masks (see *select(2)*).

A frame-buffer device can be accessed by multiple processes at once. However, each process overwrites the output of the others unless one of the lock mechanisms described here or some other synchronization mechanism is used. The lock mechanisms described here are intended for cooperating processes only.

For all frame buffers, data bytes scan from left to right and from top to bottom. A pixel, which is a visible dot on the screen, is associated with a location in the frame buffer. Each device maps one or more bits in memory to a pixel on the screen, although the bits in the frame buffer might not be continuous. Information describing the frame-buffer structure and attributes is found in the **crt_frame_buffer_t** data structure. The **crt_frame_buffer_t** data structure includes the following fields:

```
int crt_id;                /*display identifier*/
unsigned int crt_attributes; /*flags denoting attributes*/

char *crt_frame_base;      /*first byte in frame-buffer memory*/
char *crt_control_base;    /*first byte of the control*/
                           /*registers*/

char *crt_region [ CRT_MAX_REGIONS ];
                           /*other regions associated with the*/
                           /*frame-buffer device*/
```

The following are valid *ioctl(2)* requests:

GCDESCRIBE	Describe the size, characteristics, and mapped regions of the frame buffer. The information is returned to the calling process in a crt_frame_buffer_t data structure, and the parameter is defined as crt_frame_buffer_t *arg; . Although some structure fields contain addresses of one or more frame-buffer device regions, the values of these fields are not always defined. Only after a successful GCMAP command is issued (see below) are the correct addresses returned so the user can access
------------	---

the frame-buffer regions directly using the returned addresses.

GCID	Provide a device identification number. The parameter is defined as <code>int *arg;</code> . The information returned when using this command is a subset of the information provided by <code>GCDESCRIBE</code> , and is provided here for backward compatibility only.
GCON,	GCOFF Turn graphics on or off. These operations are valid for devices whose <code>CRT_GRAPHICS_ON_OFF</code> bit is set in the <code>crt_attributes</code> field of the <code>crt_frame_buffer_t</code> data structure returned by the <code>GCDESCRIBE</code> command. Otherwise, these commands have no effect.
GCAON,	GCAOFF Turn alpha on or off. These operations are valid for devices whose <code>CRT_ALPHA_ON_OFF</code> bit is set in the <code>crt_attributes</code> field of the <code>crt_frame_buffer_t</code> data structure returned by the <code>GCDESCRIBE</code> command. Otherwise, these commands have no effect.
GCMAP	Make the frame-buffer memory, graphics control, and other device regions accessible to the user process making the call. Only processes that request this can directly access frame-buffer memory and control registers. After a successful <code>GCMAP</code> call, the fields <code>crt_frame_base</code> and <code>crt_control_base</code> in the <code>crt_frame_buffer_t</code> data structure (returned by a subsequent <code>GCDESCRIBE ioctl(2)</code> call), hold the valid addresses of these two regions of the frame buffer. If, for a specific device, more than two regions are to be mapped to the user's address space, the base addresses of up to <code>CRT_MAX_REGIONS</code> extra device regions will be placed in the array <code>crt_region</code> in successive order. Only the regions pertinent to a specific frame buffer are mapped. Irrelevant region fields in the <code>crt_frame_buffer_t</code> data structure are set to 0. Use of the <code>arg</code> parameter is implementation dependent (see <code>DEPENDENCIES</code> below). The base addresses for frame-buffer regions are always page aligned.
GCUNMAP	Cause access to the frame-buffer memory, graphics control, and possibly other device regions to be removed from the requesting process. The parameter <code>arg</code> is ignored and should be set to 0. Any attempt to access these memory regions after a successful <code>GCUNMAP</code> call results in a memory fault and sends the signal <code>SIGSEGV</code> to the process.
GCLOCK	Provide for exclusive use of the frame-buffer device by cooperating processes. The calling process either locks the device and continues or is blocked. Blocking in this case means that the call returns only when the frame buffer is available or when the call is interrupted by a signal. If the call is interrupted, it returns an error and <code>errno</code> is set to <code>EINTR</code> . Waiting occurs if another process has previously locked this frame buffer using the <code>GCLOCK</code> command and has not executed a <code>GCUNLOCK</code> command yet. The <code>GCLOCK</code> command does not prevent other non-cooperating processes from writing to the frame buffer; thus, <code>GCLOCK</code> is an advisory lock only. The parameter <code>arg</code> is ignored and should be set to 0. This call prevents the Internal Terminal Emulator (ITE) from corrupting the state of the graphics hardware (see <i>termio(7)</i>). On some systems, as long as the frame buffer is locked with a <code>GCLOCK</code> command, the ITE does not output text to it (see <code>DEPENDENCIES</code> below). Any attempt to lock the device more than once by the same process fails, and causes <code>errno</code> to be set to <code>EBUSY</code> .
GCLOCK_NOWAIT	Provide for exclusive use of the frame-buffer device by cooperating processes. This request has the same effect on the frame-buffer device as does the <code>GCLOCK</code> request. However, this call does not wait for the frame buffer to be released by other processes. If the frame-buffer device is locked, the process is not blocked; instead, the system call returns an error and causes <code>errno</code> to be set to <code>EAGAIN</code> . The parameter <code>arg</code> is ignored and should be set to 0.
GCLOCK_BLOCKSIG	Provide for exclusive use of the frame-buffer device by cooperating processes while blocking all incoming signals for the calling process that otherwise might have been caught. This call is a superset of the <code>GCLOCK</code> call. The parameter <code>arg</code> is ignored and should be set to 0. When the display is acquired for exclusive use (and thus locked), all signals sent to the process that otherwise would have been caught by the process <i>at the time of the GCLOCK call</i> , are withheld (blocked) until <code>GCUNLOCK</code> is requested. Any attempt to modify the signal mask of the process (see <i>sigsetmask(2)</i>) before a <code>GCUNLOCK</code> request is made will not have any effect on these blocked signals. The signals are not blocked until the lock is actually acquired, and might be received while

still awaiting the lock.

The signal SIGTSTP is also blocked whether or not it is being caught. The signals SIGTTIN and SIGTTOU are also blocked on frame-buffer devices where the ITE does not output to the device while it is locked. See DEPENDENCIES below.

Except for the three signals mentioned above, this call does not block signals that the process did not expect to catch, nor does it block signals that cannot be caught or ignored. This command does not prevent other non-cooperating processes from writing to the frame buffer.

GCLOCK_BLOCKSIG_NOWAIT

Provide for exclusive use of the frame-buffer device by cooperating processes, while blocking all incoming signals for the calling process that otherwise would have been caught. This request has the same effect on the frame-buffer device as does the GCLOCK_BLOCKSIG request. However, this call does not wait for the frame buffer to be released by other processes. If the frame-buffer device is locked, the process is not blocked, but the system call returns an error and causes **errno** to be set to EAGAIN. The parameter *arg* is ignored and should be set to 0.

GCUNLOCK

Relinquish exclusive use of the frame-buffer device. If the device is locked with a GCLOCK_BLOCKSIG or GCLOCK_BLOCKSIG_NOWAIT *ioctl*(2) request, the signal mask of the calling process is restored to its state prior to the locking request.

GCRESET

Reset the graphic hardware associated with the frame-buffer device to a defined initial state. The call enables the frame-buffer device to respond to the *ioctl* requests defined here.

GCDMA_OUTPUT

Send DMA output to the frame-buffer device. This system call is used to transfer data from a user's array to a rectangular area of the graphics frame-buffer, or optionally, to the device's graphics control space.

The parameters for the DMA are passed in a "crt_dma_ctrl_t" data structure, which includes the following fields:

```
char *mem_addr;      /* Starting address of data
                     being transferred */
char *fb_addr;       /* Address of framebuffer
                     destination */
int length;          /* Number of bytes to transfer,
                     including those "skipped" */
int linelength;      /* Number of bytes written
                     on each framebuffer row */
int skipcount;       /* Number of source bytes to
                     ignore after each "linelength" */
unsigned int flags;  /* Specified options to the driver */
```

To write to the graphics frame-buffer, set **fb_addr** to the address of the upper-left corner of the rectangle to be drawn. The DMA will write **linelength** bytes on each frame-buffer row, ignore the next **skipcount** bytes of memory data, then resume writing at the same starting position on each succeeding frame-buffer row. This is continued until **length** bytes are either written or ignored.

To write to the graphics control space, set **fb_addr** to the address of the first graphics control register to write. In this case, **linelength** and **skipcount** are ignored.

The **flags** parameter specifies options for the DMA. Currently, there are no supported flags and this parameter should be set to zero, otherwise the system call will fail and **errno** is set to EINVAL.

The DMA has the same effect on the frame-buffer device as using store instructions to write the data. Thus, various graphics control registers may affect the results of the DMA. It is the responsibility of the user program to perform any necessary set-up of the frame-buffer device so that the DMA has the desired results.

The **skipcount** parameter allows the user to refresh a portion of a window image that the user has stored in memory for those cases where only a portion of the image needs to be refreshed. The window image is then a superset of the rectangle being

updated, and might thus have different dimensions. The `skipcount` specifies the portion of the row in the larger window image that is excluded from the rectangle. Thus, `linelength` plus `skipcount` would be the number of bytes in each row of the larger window image array.

If a particular framebuffer device supports this system call, the `CRT_DMA_OUTPUT` flag in the `crt_attributes` field of the `crt_frame_buffer_t` structure is set. Some framebuffer devices supporting DMA might restrict alignment of the various parameters, and are specified in the `DEPENDENCIES` section below. The kernel ensures that these restrictions are obeyed, and if they are not the system call will fail and set `errno` to `EINVAL`.

It is the responsibility of the application to guarantee that the system's physical memory is up-to-date by flushing the processor's data cache. One should use the `GCDMA_DATAFLUSH` ioctl to ensure that the data is consistent before initiating a DMA transfer.

GCDMA_DATAFLUSH Flush the specified data from the processor's data cache to the system's main memory. This system call is intended to be used before DMA to ensure that an up-to-date version of the data is transferred to the framebuffer or to control space.

The parameters for the flush are passed in a `crt_flush_t` data structure, which includes the following fields:

```
char *flush_addr; /* Starting address of data
                  to be flushed */
int flush_len;    /* Number of bytes to flush */
```

The kernel ensures that the `flush_len` bytes starting at `flush_addr` are consistent in main memory with respect to the cache.

GCSLOT Provide pertinent information about the calling process's participation in the system-wide graphics locking mechanism (see the discussion under `GCLOCK` above). The `GCSLOT` request does not carry out any actual locking functionality. The lock information is returned to the calling process in a `crt_gcslot_t` data structure. The parameter is defined as `crt_gcslot_t *arg`. The `crt_gcslot_t` data structure is defined in the file `<sys/framebuf.h>`.

GCSTATIC_MAP Prevent the Internal Terminal Emulator (ITE) from modifying the device's color map.

GCVARIABLE_MAP Allow the Internal Terminal Emulator (ITE) to modify the device's color map.

DEPENDENCIES

Series 700/800

When requesting `GCMAP`, the parameter *arg* is ignored and should be set to 0.

All supported ITEs ignore the frame buffer lock for output.

Series 700

Among the device identification constants that can be returned both by the `GCID` call and in the `crt_id` field of the `crt_frame_buffer_t` data structure by the `GCDESCRIBE` call are:

```
S9000_ID_98705
S9000_ID_98736
S9000_ID_A1659A
S9000_ID_A1439A
```

If a memory-mapped graphics co-processor is available, it is mapped in with other graphics regions as the result of a `GCMAP` call, and its address is recorded as the last entry in the `crt` region array returned by the `GCDESCRIBE` call.

Series 800

The following device identification constants are returned both by the `GCID` call and in the `crt_id` field of the `crt_frame_buffer_t` data structure by the `GCDESCRIBE` call:

```
S9000_ID_98720
S9000_ID_98730
S9000_ID_98550
```

For the HPA1047A Interface Card, the fields of the `crt_dma_info` structure have the following restrictions:

mem_addr	32-byte aligned
fb_addr	16-byte aligned
length	non-zero multiple of 32
skipcount	0

ERRORS

[EAGAIN]	The operation would result in suspension of the calling process, but the request was either GCLOCK_NOWAIT or GCLOCK_BLOCKSIG_NOWAIT.
[EBUSY]	Attempted to lock the device, which is already locked by the same process.
[EINTR]	A call to <i>ioctl(2)</i> was interrupted by a signal.
[EINVAL]	An invalid <i>ioctl(2)</i> command was made.
[ENODEV]	Attempted to use <i>read(2)</i> or <i>write(2)</i> system calls on the device.
[ENOMEM]	Sufficient memory for mapping could not be allocated.
[ENOSPC]	Required resources for mapping could not be allocated.
[ENXIO]	The minor number on the device file refers to a nonexistent device.
[EPERM]	Requested GCUNLOCK <i>ioctl(2)</i> command, but the device was locked by a different process.

AUTHOR

framebuf was developed by HP.

SEE ALSO

select(2), *open(2)*, *close(2)*, *signal(2)*, *sigsetmask(2)*, *lockf(2)*, *ioctl(2)*, *mknod(1M)*, *starbase(3G)*, *termio(7)*.

NAME

hil - HP-HIL device driver

SYNOPSIS

```
#include <sys/hilioctl.h>
```

DESCRIPTION

HP-HIL, the Hewlett-Packard Human Interface Link, is the Hewlett-Packard standard for interfacing a personal computer, terminal, or workstation to its input devices. **hil** supports devices such as keyboards, mice, control knobs, ID modules, button boxes, digitizers, quadrature devices, bar code readers, and touchscreens.

On systems with a single link, HP-HIL device file names use the following format:

```
/dev/hil n
```

where *n* represents a single digit that specifies the physical HP-HIL device address, which ranges from 1 to 7. For example, `/dev/hil3` is used to access the third HP-HIL device.

On systems with more than one link, HP-HIL device file names use the following format:

```
/dev/hil_m.n
```

where *m* represents the instance number, and *n* represents the physical HP-HIL device address. For example, `/dev/hil_0.2` would be used to access the second device on the link which has an instance number of zero. Likewise, `/dev/hil_12.7` references the seventh device on the link with instance number twelve.

Note that HP-HIL device addresses are determined only by the order in which devices are attached to the link. The first device attached to the link becomes device one, the second device attached becomes device two, etc.

HP-HIL devices are classified as "slow" devices. This means that system calls to **hil** can be interrupted by caught signals (see *signal(5)*).

hil can only read HP-HIL keyboards in raw keycode mode. Raw keycode mode means that all keyboard input is read unfiltered. HP-HIL keyboards return keycodes that represent key press and key release events.

Use *hilkbd(7)* to read mapped keycodes from HP-HIL keyboards. Use the Internal Terminal Emulator (ITE) described in *termio(7)* to read ASCII characters from HP-HIL keyboards.

System Calls

open(2) gives exclusive access to the specified HP-HIL device. Any previously queued input from the device is discarded. If the device is a keyboard, it is opened in raw keycode mode. A side effect of opening a keyboard in raw keycode mode is that the ITE (see *termio(7)*) and mapped keyboard driver (see *hilkbd(7)*) lose input from that keyboard until it is closed. Only device implemented auto-repeat functionality is available while in raw keycode mode (see *HILER1* and *HILER2*).

The file status flag, *O_NDELAY*, can be set to enable non-blocking reads (see *open(2)*).

close(2) returns an HP-HIL keyboard to mapped keycode mode, making its input available to the ITE or mapped keyboard driver (see *hilkbd(7)*).

read(2) returns data from the specified HP-HIL device, in time-stamped packets:

```
unsigned char packet_length;
unsigned char time_stamp[4];
unsigned char poll_record_header;
unsigned char data[ packet_length - 6 ];
```

packet_length specifies the number of bytes in the packet including itself, and can range from six to twenty bytes. *time_stamp*, when re-packed into an integer, specifies the time, in tens of milliseconds, that the system has been running since the last system boot. The most significant byte of the time stamp is *time_stamp[0]*. *poll_record_header* indicates the type and quantity of information to follow, and reports simple device status information. The number of data bytes is device dependent. Refer to the text listed in SEE ALSO for descriptions of the *poll_record_header* and device-specific data.

Usually two system calls are required to read each data packet, the first system call reads the data packet length; the second system call reads the actual data packet. Some devices always return the same amount

of data in each packet, in which case the count and the packet can both be read in the same system call.

If the file status flag, `O_NDELAY`, is set and no data is available, `read(2)` returns 0 instead of blocking.

`write(2)` is not supported by `hil`.

`select(2)` can be used to poll for available input from HP-HIL devices. `select(2)` for write or for exception conditions always returns a false indication in the file descriptor bit masks.

`ioctl(2)` is used to perform special operations on HP-HIL devices. `ioctl(2)` system calls all have the form:

```
int ioctl(int fildes, int request, char *arg);
```

The following *request* codes are defined in `<sys/hilioctl.h>`:

HILID Identify and Describe

This request returns the Identify and Describe Record in the `char` variable to which *arg* points, as supplied by the specified HP-HIL device. The Identify and Describe Record is used to determine the type and characteristics of each device connected to the link. The Identify and Describe Record can vary in length from 2 to 11 bytes. The record contains at least:

- A Device ID byte, and
- A Describe Record Header byte.

The Device ID byte is used to identify the general class of a device, and its nationality in the case of a keyboard or keypad. The Describe Record Header byte describes the position report capabilities of the device. The Describe Record Header byte also indicates if an I/O Descriptor byte follows at the end of the Describe Record. It also indicates support of the Extended Describe and the Report Security Code requests. If the device is capable of reporting any coordinates, the Describe Record contains the device resolution immediately after the Describe Record Header byte. If the device reports absolute coordinates, the maximum count for each axis is specified after the device resolution. The I/O Descriptor byte indicates how many buttons the device has. The I/O Descriptor byte also indicates device proximity detection capabilities and specifies Prompt/Acknowledge functions. All HP-HIL devices support the Identify and Describe request.

HILPST Perform Self Test

This request causes the addressed device to perform its self test, and returns the one-byte test result in the `char` variable to which *arg* points. A test result of zero indicates a successful test, non-zero results indicate device-specific failures. All HP-HIL devices support the Self Test request.

HILRR Read Register

The Read Register request expects an HP-HIL device register address in the `char` variable to which *arg* points, and returns the one-byte contents of that register in *arg*. The Extended Describe Record indicates whether a device supports the Read Register request.

HILWR Write Register

The Write Register request expects *arg* to contain a record containing one or more packets of data, each containing the HP-HIL device register address and one or more data bytes to be written to that register. There are two types of Register Writes. Type 1 can be used to write a single byte to each individual device register. Type 2 can be used to write several bytes to one register. The Extended Describe Record indicates if a device supports either or both types of register write requests.

HILRN Report Name

The Report Name request returns the device description string in the character array to which *arg* points. The string may be up to fifteen characters long. The Extended Describe Record indicates support of the Report Name request.

HILRS Report Status

The Report Status request returns the device-specific status information string in the character array to which *arg* points. The string can be up to fifteen bytes long. The Extended Describe record indicates support of the Report Status request.

HILED	Extended Describe
	The Extended Describe request returns the Extended Describe Record in the character array to which <i>arg</i> points. The Extended Describe Record may contain up to fifteen bytes of additional device information. The first byte is the Extended Describe Header, which indicates whether a device supports the Report Status, Report Name, Read Register, or Write Register requests. If the device implements the Read Register request, the maximum readable register is specified. If the device supports the Write Register request, the Extended Describe Record specifies whether the device implements either or both of the two types of register writes and the maximum writeable register. If the device supports Type 2 register writes, the maximum write buffer size is specified. The Extended Describe Record can also contain the localization (language) code for a device. Support of the Extended Describe request is indicated in the Describe Record Header byte.
HILSC	Report Security Code
	The Report Security Code request returns the Security Code Record in the character array to which <i>arg</i> points. The Security Code Record can be between one and fifteen bytes of data that uniquely identifies that particular device. Applications can use this request to implement a hardware "key" that restricts each copy of the application to a single machine or user. An application can read the Security Code Record from an HP-HIL ID Module and then verify that the application is running on a specific machine or that the application is being used by a legitimate user. Devices indicate support of the Report Security Code request in the Describe Record Header.
HILER1	Enable Auto Repeat Rate = 1/30 Second
	This request is used to enable the "repeating keys" feature implemented by the firmware of some HP-HIL keyboard and keypad devices. It also sets the cursor key repeat rate to 1/30 sec. This request does not use <i>arg</i> .
HILER2	Enable Auto Repeat Rate = 1/60 Second
	This request is used to enable the "repeating keys" feature implemented in the firmware of some HP-HIL keyboard and keypad devices. It also sets the cursor key repeat rate to 1/60 sec. This request does not use <i>arg</i> .
HILDKR	Disable Keyswitch Auto Repeat
	This request turns off the "repeating keys" feature implemented in the firmware of some HP-HIL keyboard and keypad devices. This request does not use <i>arg</i> .
HILP1..HILP7	Prompt 1 through Prompt 7
	These seven requests are supported by some HP-HIL devices to give an audio or visual response to the user, perhaps indicating that the system is ready for some type of input. A device specifies acceptance of these requests in the I/O Descriptor Byte in the Describe Record. These requests do not use <i>arg</i> .
HILP	Prompt (General Purpose)
	This request is intended as a general purpose stimulus to the user. Devices accepting this request indicate so in the I/O Descriptor Byte in the Describe Record. This request does not use <i>arg</i> .
HILA1..HILA7	Acknowledge 1 through Acknowledge 7
	These seven requests are intended to provide an audio or visual response to the user, generally to acknowledge a user's input. The I/O Descriptor Byte in the Describe Record indicates whether an HP-HIL device implements this request. These requests do not use <i>arg</i> .
HILA	Acknowledge (General Purpose)
	The Acknowledge request is intended to provide an audio or visual response to the user. Devices accepting this request indicate so in the I/O Descriptor Byte in the Describe Record. This request does not use <i>arg</i> .

ERRORS

[EBUSY]	The specified HP-HIL device is already opened.
[EFAULT]	A bad address was detected while attempting to use an argument to a system call.
[EINTR]	A signal interrupted an <i>open(2)</i> , <i>read(2)</i> , or <i>ioctl(2)</i> system call.
[EINVAL]	An invalid parameter was detected by <i>ioctl(2)</i> .
[ENXIO]	No device is present at the specified address; see WARNINGS, below.
[EIO]	A hardware or software error occurred while executing an <i>ioctl(2)</i> system call.
[ENODEV]	<i>write(2)</i> is not implemented for HP-HIL devices.

WARNINGS

An ENXIO error is returned by *open(2)* and *ioctl(2)* if any attempt is made to access a device while **hil** is reconfiguring the link during power-failure recovery.

hil cannot detect whether or not a device executed an *ioctl(2)* request.

HP-HIL devices have no status bit available to indicate whether they support the HILER1, HILER2, or HILDKR requests.

AUTHOR

hil was developed by the Hewlett-Packard Company.

FILES

`/dev/hil[1-7]`
`/dev/hil_*. [1-7]`

SEE ALSO

close(2), *errno(2)*, *fcntl(2)*, *ioctl(2)*, *open(2)*, *read(2)*, *select(2)*, *signal(5)*, *hilkbd(7)*, *termio(7)*.

For detailed information about HP-HIL hardware and software in general, see the *HP-HIL Technical Reference Manual*.

NAME

hilkbd - HP-HIL mapped keyboard driver

DESCRIPTION

HP-HIL, the Hewlett-Packard Human Interface Link, is the Hewlett-Packard standard for interfacing a personal computer, terminal, or workstation to its input devices. *hilkbd* supplies input from all mapped keyboards on a specified HP-HIL link.

hilkbd returns mapped keycodes, not ASCII characters. "Raw" keycodes are the individual key downstrokes and upstrokes, and are different for each type of keyboard. *hilkbd* maps the raw input into the keycodes and protocol expected by the HP-UX, Pascal Workstation, and BASIC/UX operating systems. The *hil*(7) driver can usurp a keyboard from *hilkbd* by changing it from mapped mode to raw mode.

System Calls

open(2) gives exclusive access to the keyboard. If there is an ITE (internal terminal emulator) associated with the keyboard, the ITE loses input from the keyboard until the keyboard device is closed. Any previous queued input for the keyboard device is flushed from the input queue.

close(2) returns control of the keyboard to the ITE, if present. Any unread input is discarded at that time.

read(2) returns data from the keyboard in time-stamped packets:

```
unsigned char time_stamp[4];
unsigned char status;
unsigned char data;
```

time_stamp, when repacked into an integer data type of four or more bytes, specifies the time since an arbitrary point in the past (e.g., system start-up time). This point does not change between packets, but time during a power failure may or may not be counted. The time is in units of tens of milliseconds.

The *status* byte encodes the state of the keyboard **Shift** and **Ctrl** keys:

```
0x8X  shift and control
0x9X  control only
0xAX  shift only
0xBX  no shift or control
```

The *data* byte contains the actual keystroke.

If the file status flag `O_NDELAY` is set, *read*(2) returns **0** instead of blocking, when no data is available. The *read*(2) system call on an HP-HIL keyboard is considered "slow"; that is, it can be interrupted by caught signals (see *signal*(2)).

write(2) is not supported by *hilkbd*.

select(2) can be used to poll for input to read from *hilkbd* devices. *select*(2) for write or for exceptional conditions always returns a false indication in the bit masks.

ioctl(2) is used to perform special operations on the device. *ioctl*(2) system calls have the form:

```
int ioctl(int fildes, int request, char *arg);
```

The following *hilkbd* request codes are defined in `<sys/hilioctl.h>`:

`KBD_READ_CONFIG`

Read the configuration code.

This request returns a one-byte configuration code in the **char** variable to which *arg* points. This contains a field, defined by `KBD_IDCODE_MASK`, which specifies the keyboard identification code. The possible values of this field are defined in the header file, and this identification code affects interpretation of the language code. All other fields in the configuration code are currently undefined.

`KBD_READ_LANGUAGE`

Read the language code.

This request returns a one-byte language code, as read from the keyboard, in the **char** variable to which *arg* points. If there is more than one keyboard, the language is taken from the first keyboard on the link. Interpretation of the language code is affected by the keyboard identification field within the configuration code.

KBD_STATUS Read the keyboard status register.

This request returns a one-byte value containing bit flags specifying the state of the shift and control keys in the **char** variable to which *arg* points:

KBD_STAT_LEFTSHIFT	The left shift key is up
KBD_STAT_RIGHTSHIFT	The right shift key is up
KBD_STAT_SHIFT	Both shift keys are up
KBD_STAT_CTRL	The control key is up

Other bits are undefined.

KBD_REPEAT_RATE

Set the keyboard auto-repeat rate.

The one-byte value to which *arg* points is the negative of the repeat period, in tens of milliseconds. The repeat rate is the reciprocal of the repeat period. A parameter of zero disables auto-repeat.

KBD_REPEAT_DELAY

Set the keyboard auto-repeat delay.

The one-byte value to which *arg* points is the negative of the repeat delay, in tens of milliseconds.

KBD_BEEP

Cause an audible beep.

The one-byte value to which *arg* points specifies the volume of the beep, within the range 0 through KBD_MAXVOLUME. Implementations with fewer than KBD_MAXVOLUME discrete levels of volume will scale the parameter into the smaller range.

ERRORS

[EINVAL]	An invalid parameter was detected by <i>ioctl(2)</i> .
[EINTR]	A signal was caught during a <i>read(2)</i> system call.
[ENXIO]	No keyboard is present on the HP-HIL link specified by the minor number.
[ENODEV]	An attempt was made to use <i>write(2)</i> using <i>hilkbd</i> .
[EBUSY]	The device is already open.

AUTHOR

hilkbd was developed by the Hewlett-Packard Company.

FILES

/dev/hilkbd*

SEE ALSO

termio(7), *hil(7)*, *mknod(1M)*, *select(2)*, *signal(2)*.

NAME

hplib - Hewlett-Packard Interface Bus driver (OBSOLETE AT 10.30)

SYNOPSIS

```
#include <sys/hplibio.h>
```

DESCRIPTION

HP-IB is Hewlett-Packard's implementation of the Institute of Electrical and Electronic Engineers Standard Digital Interface for Programmable Instrumentation (IEEE Std 488-1978). This section describes the use of the HP-IB driver in the HP-UX system.

Auto-addressed Files vs. Raw Bus Files

A major distinction is made in the HP-UX driver between "auto-addressed" files and "raw bus" files. An auto-addressed file is associated with a specified address on the HP-IB. The user need not be concerned with any HP-IB addressing or commands; the driver handles device addressing and unaddressing during data transfers. However, the user is limited to transactions to and from a single device. A raw bus file, on the other hand, gives the user access to the entire HP-IB; responsibility for all commands and addressing lies with the user. The raw bus file is typically used to access multiple devices on the same bus, as well as provide universal device commands such as interface clear and parallel poll.

Although differences exist between auto-addressed and raw bus files, the user/driver interface is consistent across both types. Therefore, each category of requests is presented with separate subsections for auto-addressed and raw bus files.

Naming Convention

HP-IB device files are named according to the following format:

```
/dev/hplib/c#[t#d0]
```

where **c#** specifies the card instance of the bus, **t#** specifies the HP-IB address on that bus, and **d0** specifies a device unit of zero. A device file specifying only the card instance denotes the raw bus. Files with the address suffix are auto-addressed.

Device Attributes

HP-IB attributes are classified into two groups, per-open and per-interface. File descriptors obtained from separate **open()** requests have separate per-open attributes (see *open(2)*); changing an attribute from the per-open group affects requests on that file descriptor only. Attributes in the per-interface group are shared by all file descriptors on that interface; changing an attribute from one file descriptor affects all users of the interface.

The per-open set of attributes and the driver requests to change them are listed in the following table. All other attributes are per-interface.

Attribute	Driver Request
timeout	HPIB_TIMEOUT
write termination mode	HPIB_EOI
read termination pattern	HPIB_READ_PATTERN
read termination reason	HPIB_TERM_REASON
signal mask	HPIB_SIGNAL_MASK
lock count	HPIB_LOCK
wait events	HPIB_WAIT_ON_STATUS

Transfer Requests

The standard **read()** and **write()** requests are used for data transfer over HP-IB (see *read(2)* and *write(2)*). However, their actions are slightly different for each type of file. Raw bus files place data directly onto the bus. No addressing or unaddressing of devices is done by the driver; this is the user's responsibility.

On the other hand, the driver does all addressing for transactions with auto-addressed files. The actual sequence of events is:

```
UNL, <device addressing>, <data>, <terminator>
```

All write requests end when the specified number of bytes has been transferred over the bus. Optionally, the HP-IB **END** message can be sent with the last byte written; this is controlled via the **HPIB_EOI** request. All read requests end when the specified number of bytes has been read over the bus or when the

(Series 800 Only)

device asserts **EOI**. In addition, a single character can be designated to end the read operation via the **HPIB_READ_PATTERN** request.

Control Requests

Control requests cause some action on the bus. All such requests have the same format:

```
struct io_ctl_status {
    int type;
    int arg[3];
} hpib_control;

ioctl(fildes, IO_CONTROL, &hpib_control);
```

In the **io_ctl_status** structure, the **type** field specifies the type of control function required, while the **arg** array holds any associated arguments. The defined values for **type** and their use are described as follows:

HPIB_TIMEOUT	Set the timeout. If any transaction for this file takes longer than arg[0] microseconds, it is aborted with a status of ETIMEDOUT returned to the user. This is used mainly for detecting device failure. A timeout of 0 is equivalent to infinity; that is, no transaction will time out.
HPIB_WIDTH	Set the width of the interface. This request specifies the number of valid data lines on transfers; arg[0] holds the desired interface width in bits. All future read requests inspect only the least significant arg[0] data lines, and all future writes present data on only those lines. The state of all other data lines is indeterminate.
HPIB_SPEED	Set the transfer speed of the interface. The desired data transfer speed in kilobytes per second is specified in arg[0] . Note that this value is advisory only, and is typically used by the driver to determine the method of data transfer.
HPIB_EOI	Enable/disable EOI assertion on writes. If arg[0] is nonzero, all subsequent writes end with EOI asserted on the last byte transferred. A zero arg[0] disables EOI assertion.
HPIB_SYSTEM_CTLR	Make the interface system controller or non-system controller. If arg[0] is nonzero, the interface becomes the system controller. A zero in arg[0] sets the interface to non-system controller. This request is applicable to raw bus files only.
HPIB_READ_PATTERN	Enable or disable pattern matching on reads. If arg[0] is nonzero, all subsequent reads terminate when the pattern specified in arg[1] is encountered in the input stream. This termination condition is subject to all other termination conditions in effect for the file. Only the <i>n</i> least significant bits of the pattern are used in the match, where <i>n</i> is the interface's current width, set via HPIB_WIDTH . A zero arg[0] disables read pattern matching.
HPIB_SIGNAL_MASK	Define signaling events. This request allows the calling process to receive a signal when some event occurs on the HP-IB. The event or events are specified by computing the bitwise inclusive OR of the values from the list below, and placing the mask in arg[0] . All of these events can be enabled on raw bus files, but only ST_SRQ and ST_PPOLL apply to auto-addressed files.
ST_SRQ	Signal on assertion of Service Request (SRQ).
ST_PPOLL	Signal when device responds to Parallel Poll.
ST_REN	Signal when interface enters remote state.
ST_ACTIVE_CTLR	Signal when interface becomes active controller.
ST_TALK	Signal when interface is addressed to talk.
ST_LISTEN	Signal when interface is addressed to listen.
ST_IFC	Signal on assertion of Interface Clear (IFC).
ST_DCL	Signal on receipt of Device Clear (DCL).

(Series 800 Only)

ST_GET

Signal on receipt of Group Execute Trigger (GET).

When any subsequent flagged event occurs, the process is sent **SIGEMT**. The user should set up a handler to trap this signal via **signal()** or **sigvector()** (see *signal(2)* or *sigvector(2)*). The reason for interrupt can be obtained via the **IO_STATUS** request **HPIB_SIGNAL_MASK**. Each request overwrites the previous mask for the file; therefore events can be disabled by using a zero **arg[0]**.

If **ST_PPOLL** is flagged, the user supplies additional information in the **arg** array. For raw bus files, the low-order bytes of **arg[1]** and **arg[2]** contain eight-bit masks with each bit corresponding to a Data I/O (DIO) line and the least significant bit mapped to DIO1. When a device responds to parallel poll, it asserts the appropriate line; **arg[1]**'s bits indicate the parallel poll sense of this assertion. Bits set in **arg[2]** indicate that the corresponding address is capable of responding to polling. For auto-addressed files, **arg[1]** specifies the parallel poll sense of the assigned device's response to parallel poll. Parallel poll interrupts can be enabled only if the interface is the active controller.

HPIB_LOCK

Lock or unlock the HP-IB interface. Setting **arg[0]** to **LOCK_INTERFACE** locks the HP-IB interface, giving the calling process exclusive access to the card and bus. The lock is incremental; that is, if the interface is already locked by the current process, additional lock requests increment a per-open lock count maintained in the driver.

An **arg[0]** of **UNLOCK_INTERFACE** decrements the per-open lock count; when the total interface lock count drops to zero, the lock is cleared. The lock can also be cleared by setting **arg[0]** to **CLEAR_ALL_LOCKS**, which removes all locks held by the current process.

After a successful lock or unlock, **arg[1]** contains the current lock count for this open, and **arg[2]** contains the total lock count on this interface.

While the interface is locked, other processes that attempt to access the bus or interface are blocked until either the interface becomes unlocked or the process's per-open timeout (set via **HPIB_TIMEOUT**) expires. However, if the **O_NDELAY** file status flag is set (see *fcntl(5)*), the user request fails and returns immediately with the **EACCES** error. See the Summary of Privilege Requirements section for a list of user requests that might block.

HPIB_ADDRESS

Set the HP-IB address to which the interface responds when it is not the active controller. The bus address is set via **arg[0]**, and must be between 0 and 30 decimal. Two additional flags, **HPIB_TALK_ALWAYS** and **HPIB_LISTEN_ALWAYS**, can be set by computing the bitwise inclusive OR of their values with the address. These flags enable the interface to talk, and/or listen always, respectively. This request is applicable to raw bus files only.

HPIB_RESET

Reset the device or bus, depending on which of the following values is in **arg[0]**:

DEVICE_CLR	Address the device and send a selective device clear (SDC) command. This applies only to auto-addressed files.
BUS_CLR	Assert Interface Clear (IFC) and Remote Enable (REN), and clear Attention (ATN).
HW_CLR	Reset bus interface card. The card is self-tested and if the card is system controller, IFC is pulsed. All other card state information is preserved. This applies to raw bus files only.

HPIB_PPOLL_RESP

Control the interface's response to parallel poll. When the interface is not acting as active controller, it can be enabled to respond to parallel polling by the current active controller. If **arg[0]** is nonzero, the interface responds to parallel poll. **arg[1]** specifies the DIO line on which the card responds; **arg[1]** has a value between 0 and 7, with a value of 0 mapping to DIO1, 1 mapping to DIO2, and so forth. The parallel poll sense of the response is determined by **arg[2]**. An **arg[0]** of zero disables the interface's response to parallel poll.

(Series 800 Only)

For auto-addressed files, the file's associated device address is configured, rather than the interface.

HPIB_PPOLL_IST

Enable or disable response to parallel poll. If `arg[0]` is nonzero, the interface responds to parallel poll. An `arg[0]` of zero disables the interface's response. This differs from the previous request, because the parallel poll sense and address of the interface's response are unchanged. This request applies to raw bus files only.

HPIB_REN

Place a device into or out of the remote state. For a raw bus file, this request merely sets or clears the Remote Enable line, depending on whether `arg[0]` is nonzero or zero respectively. For auto-addressed files, a nonzero `arg[0]` asserts the Remote Enable line and addresses the device. If `arg[0]` is zero, the device is removed from the remote state by sending it a Go-to-Local command (GTL).

HPIB_SRQ

Request service. This request causes the interface to assert the Service Request line (SRQ) until it is serially polled. At that time it responds with the status byte given in `arg[1]`. This request applies to raw bus files only.

This request is normally used only when the interface is not the active controller. Nonetheless, the active controller can assert SRQ, and the **HPIB_BUS_STATUS** request will reflect the assertion; however, the SRQ line does not change state until the interface passes control.

HPIB_PASS_CONTROL

Pass active control of the bus. If the interface is currently active controller, this request relinquishes control of the bus, passing it instead to the device at the bus address in `arg[0]`. Passing control should be done with care, since it is not possible to detect whether the named device can indeed assume bus control. This request applies only to raw bus files.

HPIB_GET_CONTROL

Become active controller. This request causes the interface to assert Interface Clear (IFC) and Remote Enable (REN) as a means of regaining control of the HP-IB. It applies only to raw bus files.

Transparent Bus Request

This request allows a user to send direct commands over the HP-IB; it should be used with care, since improper use might place the bus in an unusable state.

The transparent bus request takes the following form:

```
struct hpib_command {
    int length;
    char buffer[MAX_HPIB_COMMANDS];
} hpib_cmd;

ioctl(fildes, HPIB_COMMAND, &hpib_cmd);
```

This call transmits *length* bytes of data in *buffer* over the HP-IB with Attention (ATN) asserted. On completion of the request, ATN remains asserted.

For commands sent through an auto-addressed file, *buffer* is surrounded with the appropriate device addressing. What appears on the bus is:

UNL, TALK CIC, LISTEN device, *buffer*

This differs from the approach toward a raw bus file. For such files, the *buffer* is merely placed on the bus with ATN asserted, with no addressing or unaddressing.

Status Requests

These requests are used to obtain information about the general state of a device or the HP-IB. Their calling sequence is similar to that of control requests:

```
struct io_ctl_status hpib_status;

ioctl(fildes, IO_STATUS, &hpib_status);
```

As with the data structure to control requests, the `type` field specifies the type of information requested, while the `arg` array holds clarification data. The defined status requests for HP-IB and their use are

(Series 800 Only)

described as follows:

HPIB_ADDRESS	Return the bus address associated with the file in arg[0] .																		
HPIB_TIMEOUT	Return the interface's timeout in microseconds in arg[0] .																		
HPIB_WIDTH	Return the interface's path width in bits in arg[0] .																		
HPIB_SPEED	Return the interface's data transfer rate in K-bytes per second in <i>arg[0]</i> .																		
HPIB_READ_PATTERN	Return the interface's read termination pattern in arg[0] ; if pattern matching is not enabled, arg[0] holds a -1.																		
HPIB_SIGNAL_MASK	Return the reason for the last signal. This request returns a mask in arg[0] , with bits set indicating the reason(s) for the last SIGEMT sent to the user process. Bit definitions are identical to those of the corresponding IO_CONTROL request.																		
HPIB_LOCK	Return lock status. If the device is locked to a process, return that process ID in arg[0] and the interface lock count in arg[1] . If the device is not locked, arg[0] holds a -1.																		
HPIB_TERM_REASON	Return end conditions for the last read from this device or bus. This request returns a byte in arg[0] , with a mask of reason(s) for the completion of the last read from the device or raw bus. Applicable bits are: <table> <tr> <td>TR_COUNT</td><td>Read requested number of bytes.</td></tr> <tr> <td>TR_MATCH</td><td>Detected specified match pattern.</td></tr> <tr> <td>TR_TIMEOUT</td><td>Timed out.</td></tr> <tr> <td>TR_END</td><td>Device asserted EOI.</td></tr> <tr> <td>TR_ERROR</td><td>Detected bus error.</td></tr> <tr> <td>TR_NOTERM</td><td>No read done since open.</td></tr> </table>	TR_COUNT	Read requested number of bytes.	TR_MATCH	Detected specified match pattern.	TR_TIMEOUT	Timed out.	TR_END	Device asserted EOI.	TR_ERROR	Detected bus error.	TR_NOTERM	No read done since open.						
TR_COUNT	Read requested number of bytes.																		
TR_MATCH	Detected specified match pattern.																		
TR_TIMEOUT	Timed out.																		
TR_END	Device asserted EOI.																		
TR_ERROR	Detected bus error.																		
TR_NOTERM	No read done since open.																		
HPIB_PPOLL	Conduct a parallel poll. This request returns the bus response to parallel poll in the least significant byte of arg[0] , with DIO1 corresponding to the least significant bit. The driver delays at least 100 microseconds before reading the poll response, thus allowing the use of HPIB_PPOLL on systems with extended buses. This request applies to both auto-addressed and raw bus files.																		
HPIB_S POLL	Conduct a serial poll. For raw bus files, this request conducts a serial poll of the device address in arg[1] ; the status byte returned by the device is available in arg[0] . Auto-addressed files ignore any address in arg[1] , polling instead the device's predefined address.																		
HPIB_BUS_STATUS	Return the status of the HP-IB. This request, applicable to both types of files, returns information related to the current bus state. On return, arg[0] holds a value with bits set indicating: <table> <tr> <td>ST_NDAC</td><td>NDAC is being asserted.</td></tr> <tr> <td>ST_SRQ</td><td>SRQ is being asserted.</td></tr> <tr> <td>ST_REN</td><td>Interface is in the remote state.</td></tr> <tr> <td>ST_ACTIVE_CTLR</td><td>Interface is active controller.</td></tr> <tr> <td>ST_SYSTEM_CTLR</td><td>Interface is system controller.</td></tr> <tr> <td>ST_TALK</td><td>Interface is addressed to talk.</td></tr> <tr> <td>ST_LISTEN</td><td>Interface is addressed to listen.</td></tr> <tr> <td>ST_TALK_ALWAYS</td><td>Interface is configured to talk always.</td></tr> <tr> <td>ST_LISTEN_ALWAYS</td><td>Interface is configured to listen always.</td></tr> </table>	ST_NDAC	NDAC is being asserted.	ST_SRQ	SRQ is being asserted.	ST_REN	Interface is in the remote state.	ST_ACTIVE_CTLR	Interface is active controller.	ST_SYSTEM_CTLR	Interface is system controller.	ST_TALK	Interface is addressed to talk.	ST_LISTEN	Interface is addressed to listen.	ST_TALK_ALWAYS	Interface is configured to talk always.	ST_LISTEN_ALWAYS	Interface is configured to listen always.
ST_NDAC	NDAC is being asserted.																		
ST_SRQ	SRQ is being asserted.																		
ST_REN	Interface is in the remote state.																		
ST_ACTIVE_CTLR	Interface is active controller.																		
ST_SYSTEM_CTLR	Interface is system controller.																		
ST_TALK	Interface is addressed to talk.																		
ST_LISTEN	Interface is addressed to listen.																		
ST_TALK_ALWAYS	Interface is configured to talk always.																		
ST_LISTEN_ALWAYS	Interface is configured to listen always.																		

(Series 800 Only)

HPIB_WAIT_ON_PPOLL

Wait (sleep) until a given device responds to parallel poll. This request blocks the user until either the user's device responds to parallel poll (for auto-addressed files) or until any enabled devices respond (for raw bus files).

For a raw bus file, **arg[1]** and **arg[2]** contain eight-bit masks as defined in the **HPIB_SIGNAL_MASK** request. The return value of the request in **arg[0]** shows which devices responded to parallel poll.

For an auto-addressed file, **arg[1]** specifies the sense of the particular device's assertion. Successful completion of the request implies that the device responded.

HPIB_WAIT_ON_STATUS

Wait (sleep) until any of a set of given states is entered. The event(s) to await are specified by computing the bitwise inclusive OR of the values from the list below, and placing the mask in **arg[0]**. Applicable bits are:

ST_SRQ	Wait until SRQ is asserted.
ST_ACTIVE_CTLR	Wait until user is active controller.
ST_TALK	Wait until user is addressed to talk.
ST_LISTEN	Wait until user is addressed to listen.

Note that more than one bit can be set, thereby waiting for any of the events to occur. The return value in **arg[0]** is modified to show the actual event(s) that ended the wait. This is applicable to raw bus files only.

HPIB_INTERFACE_TYPE

Return the interface type. This returns one of two values in **arg[0]**:

HPIB_INTERFACE	The open file is a HP-IB raw bus file.
HPIB_DEVICE	The open file is a HP-IB auto-addressed file.

Extended Status Request

If the user wants to obtain several status variables in one request, the following request can be used:

```
struct io_environment hpib_env;
ioctl(fildes, IO_ENVIRONMENT, &hpib_env);
```

where the **io_environment** structure includes the following fields:

```
int interface_type;
int timeout;
int status;
int term_reason;
int read_pattern;
int signal_mask;
int width;
int speed;
int locking_pid;
```

Summary of Privilege Requirements

The following table summarizes which **ioctl()** requests can be performed under what circumstances (see **ioctl(2)**). The first three columns indicate whether the interface must be in the controlling state given to perform the request. An entry of Y means that the interface *must* be in that state, N means that the interface must *not* be in that state, and – means that the state is irrelevant. The next two columns indicate whether the request works for auto-addressed or raw bus files. The final column indicates whether the request is subject to blocking while the interface is locked (see **HPIB_LOCK**).

If an entry is marked with an asterisk (*), check the particular request for more information.

(Series 800 Only)

Request	Non-Ctrl	Active Ctrl	System Ctrl	Auto Addr	Raw Bus	Lock Enforced
IO_CONTROL						
HPIB_TIMEOUT	–	–	–	Y	Y	N
HPIB_WIDTH	–	–	–	Y	Y	Y
HPIB_SPEED	–	–	–	N	Y	Y
HPIB_EOI	–	–	–	Y	Y	N
HPIB_SYSTEM_CTRL	–	–	–	N	Y	Y
HPIB_READ_PATTERN	–	–	–	Y	Y	N
HPIB_SIGNAL_MASK	*	*	–	Y	Y	Y
HPIB_LOCK	–	–	–	Y	Y	Y
HPIB_ADDRESS	–	–	–	N	Y	Y
HPIB_RESET						
DEVICE_CLR	N	Y	–	Y	N	Y
BUS_CLR	–	–	Y	Y	Y	Y
HW_CLR	–	–	–	N	Y	Y
HPIB_PPOLL_RESP	N	Y	–	Y	Y	Y
HPIB_PPOLL_IST	–	–	–	N	Y	Y
HPIB_REN	–	–	Y	Y	Y	Y
HPIB_SRQ	–	*	–	N	Y	Y
HPIB_PASS_CONTROL	N	Y	–	N	Y	Y
HPIB_GET_CONTROL	–	–	Y	N	Y	Y
HPIB_COMMAND	N	Y	–	Y	Y	Y
IO_STATUS						
HPIB_ADDRESS	–	–	–	Y	Y	Y
HPIB_TIMEOUT	–	–	–	Y	Y	N
HPIB_WIDTH	–	–	–	Y	Y	N
HPIB_SPEED	–	–	–	Y	Y	N
HPIB_READ_PATTERN	–	–	–	Y	Y	N
HPIB_SIGNAL_MASK	–	–	–	Y	Y	N
HPIB_LOCK	–	–	–	Y	Y	N
HPIB_TERM_REASON	–	–	–	Y	Y	N
HPIB_PPOLL	N	Y	–	Y	Y	Y
HPIB_S POLL	N	Y	–	Y	Y	Y
HPIB_BUS_STATUS	–	–	–	Y	Y	Y
HPIB_WAIT_ON_PPOLL	N	Y	–	Y	Y	Y
HPIB_WAIT_ON_STATUS	–	–	–	N	Y	Y
HPIB_INTERFACE_TYPE	–	–	–	Y	Y	N
IO_ENVIRONMENT	–	–	–	Y	Y	Y

Default Configuration

The default configuration of any HP-IB file is:

```

Timeout          Infinite
Path Width       8 bits
Transfer Speed   0
EOI              Assertion Enabled
Pattern Match    Disabled
Enabled Signals  None
Locking          Unlocked
Termination Reason TR_NOTERM

```

ERRORS

A –1 return value for a driver request indicates that an error occurred; **errno** is set to specify the reason. In addition to errors defined in *open(2)*, *close(2)*, *read(2)*, *write(2)*, and *ioctl(2)*, a driver request can fail if any of the following conditions are encountered:

[EACCES] The interface is not in the active-controller or system-controller state.

(Series 800 Only)

[EACCES]	The interface is currently locked by another process via HPIB_LOCK .
[EACCES]	A request to access the file would block and the O_NDELAY file status flag is set for the file descriptor.
[EINVAL]	The request is not applicable to this type of file. Alternatively, <i>type</i> or <i>arg</i> has an invalid value.
[EINTR]	An interface power failure occurred during the processing of this request; the device might have lost state.
[EINTR]	A signal was received either while waiting for the interface to become unlocked, or while waiting for a HPIB_WAIT_ON_PPOLL or HPIB_WAIT_ON_STATUS request.
[EIO]	An unclassified error occurred.
[EMFILE]	The number of simultaneous open() requests on this interface exceeds the maximum allowed.
[ENXIO]	No bus interface is associated with the device file.
[EPERM]	An attempt was made to unlock an interface that was not locked.
[ERANGE]	The interface lock count was exceeded.
[ETIMEDOUT]	The transaction did not complete within the timeout specified.

WARNINGS

It is possible to circumvent the bus protection mechanisms afforded by the auto-addressed and raw bus dichotomy. Specifically, a user of an auto-addressed file can send commands to any or all devices on the bus with the **HPIB_COMMAND** request, if the proper device addressing is done within the data buffer.

The **HPIB_LOCK** request should be used with care. Since it provides an exclusive lock, invoking the **HPIB_LOCK** blocks access to any system disk or swap device on the associated bus.

Processes that use **HPIB_LOCK** should clear all locks before exiting. The driver attempts to clear them if the process terminates unexpectedly; however, a lock might be left outstanding if the locker dies after creating new file descriptors (via **fork()** or **dup()** that refer to the same device file (see *fork(2)* or *dup(2)*). Ensuring that all open file descriptors on a given interface are closed remedies the situation.

By default, some HP-IB peripherals respond to parallel poll on **DIO_n**, where *n* has the value of 8 minus the device's bus address. That is, a device at address 6 can respond on **DIO2**. Therefore, the results of an **HPIB_PPOLL** request can be misleading if some devices are not remotely configured.

It is impossible to transfer data using a secondary address in a single driver request.

DEPENDENCIES**HP 27110B**

The **HPIB_SPEED** and **HPIB_SYSTEM_CTLR** requests are not supported; they are configured by switches on the device adapter.

The **HPIB_SRQ** request can affect only the RQS bit of the serial poll response byte; all other bits are masked to zero by the hardware.

AUTHOR

hpiib was developed by HP.

FILES

/dev/hpiib/*

SEE ALSO

fcntl(5), **ioctl(2)**, **signal(2)**, **sigvector(2)**, specific device documentation in section 7.

NAME

inet - Internet protocol family

SYNOPSIS

```
#include <sys/types.h>
#include <netinet/in.h>
```

DESCRIPTION

The internet protocol family is a collection of protocols layered on top of the **Internet Protocol** (IP) network layer, which utilizes the internet address format. The internet family supports the SOCK_STREAM and SOCK_DGRAM socket types.

Addressing

Internet addresses are four byte entities. The include file `<netinet/in.h>` defines this address as the structure `struct in_addr`.

Sockets bound to the internet protocol family utilize an addressing structure called `struct sockaddr_in`. Pointers to this structure can be used in system calls wherever they ask for a pointer to a `struct sockaddr`.

There are three fields of interest within this structure. The first is `sin_family`, which must be set to AF_INET. The next is `sin_port`, which specifies the port number to be used on the desired host. The third is `sin_addr`, which is of type `struct in_addr`, and specifies the address of the desired host.

Protocols

The internet protocol family is comprised of the IP network protocol, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). TCP is used to support the SOCK_STREAM socket type while UDP is used to support the SOCK_DGRAM socket type. The ICMP message protocol and IP network protocol are not directly accessible.

The local port address is selected from independent domains for TCP and UDP sockets. This means that creating a TCP socket and binding it to local port number 10000, for example, does not interfere with creating a UDP socket and also binding it to local port number 10000 at the same time.

Port numbers in the range 1-1023 inclusive are reserved for use by the super-user only. Attempts to bind to port numbers in this range by non-super-users fail and result in an error returned.

AUTHOR

inet was developed by the University of California, Berkeley.

SEE ALSO

tcp(7P), udp(7P).

NAME

iomap - physical I/O address mapping

SYNOPSIS

```
#include <sys/iomap.h>
```

DESCRIPTION

The **iomap** mechanism allows the mapping (thus direct access) of physical I/O addresses into the user process address space. For PA-RISC machines, the physical I/O address space begins at **0xf0000000** and extends to **0xffffffff**.

The special (device) files for **iomap** devices are character special files using the dynamic major number allocation scheme.

The minor number for **iomap** devices is of the form:

```
0xAAAASM
```

The physical I/O address is formed by prefixing **0xAAAA** with **0xF**, and by appending **0x000** (this forces the I/O address to be page-aligned). The size of the region to be mapped is given by the expression $M \cdot (2^S)$ 4K pages. For example, the minor number for a device starting at **0xf4000000** that occupies 64MB is **0x4000e1**.

The **iomap** driver must be explicitly added to the **/stand/system** file, the kernel rebuilt, and the system subsequently rebooted prior to first using **iomap**.

I/O space is always mapped with both read and write access rights, regardless of the actual permissions on the device special file.

Multiple processes can have concurrently a single **iomap** device opened and mapped. It is the responsibility of the processes to synchronize their access.

Successive calls to **iomap** to map the same I/O space must be identical to the first mapping. Identical mappings have the same address and size.

Note that a process can additionally share I/O space (mapped by **iomap**) with a kernel driver. However, this is only possible if the driver maps in the I/O space with user read/write access rights using the appropriate driver I/O mapping services. Any I/O space mapped by drivers with kernel read/write access rights cannot be concurrently mapped by processes using **iomap**.

No **read()** or **write()** system calls are supported by the **iomap** driver.

The **ioctl()** function is used to control the **iomap** device. The following **ioctl()** requests are defined in **<iomap.h>**:

IOMAPMAP

Map the **iomap** device into user address space at the location specified by the pointer to which the **(void **)** third argument to **ioctl()** points. If the argument points to a variable containing a null pointer, the system selects an appropriate address. **ioctl()** then returns the user address where the device was mapped, storing it at the address pointed to by the third argument (see **EXAMPLES** below). Multiple processes can concurrently have the same **iomap** device mapped.

IOMAPUNMAP

Unmap the **iomap** device from the user address space.

close() shuts down the file descriptor associated with the **iomap** device. If the close is for the last system wide open on the device, the **iomap** device is also unmapped from the user address space; otherwise it is left mapped into the user address space (see **IOMAPUNMAP** above).

WARNING

Be extremely careful when creating and using **iomap** devices. Inappropriate accesses to I/O devices or RAM can result in a system crash.

ERRORS

- | | |
|----------|---|
| [EINVAL] | The address field was out of range, or the ioctl request was invalid. |
| [ENOMEM] | Not enough memory could be allocated for the mapping. |
| [EBUSY] | Device was already mapped and this mapping was not identical to the initial mapping (same address, size and access rights). |

[ENODEV]	Read and write calls are unsupported.
[ENXIO]	No such device at the address specified by the minor number.
[ENOSPC]	Required resources for mapping could not be allocated.
[ENOTTY]	Inappropriate <code>ioctl</code> request for this device type; <i>fildev</i> is not a file descriptor for an <code>iomap</code> device file.

EXAMPLES

Consider the following code fragment:

```
#include <sys/iomap.h>
...
int fildev;
void *addr;
...
addr = REQUESTED_ADDRESS;
(void) ioctl(fildev, IOMAPMAP, &addr);
(void) printf("actual address = 0x%x\n", addr);
```

where `fildev` is an open file descriptor for the device special file and `REQUESTED_ADDRESS` is the address originally requested by the program.

If `addr` is a null pointer, the system selects a suitable address then returns the selected address in `addr`.

If the value in `addr` is not a null pointer, it is used as a specified address for allocating memory. If the specified address cannot be used, an error is returned (see `ERRORS`).

SEE ALSO

`mknod(1M)`.

NAME

IP - Internet Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);
```

DESCRIPTION

IP is the network-layer protocol used by the Internet protocol family. It encapsulates TCP and UDP messages into datagrams to be transmitted by the network interface. Normally, applications do not need to interface directly to IP. However, certain multicast socket options are controlled by passing options to the IPPROTO_IP protocol level through a UDP socket, and IP Type of Service is controlled by passing an option to the IPPROTO_IP protocol level through either a TCP or UDP socket. (See the *getsockopt(2)* manual page.)

The following socket options are defined in the include file `<netinet/in.h>`. The type of the variable pointed to by the *optval* parameter is indicated in parentheses. The data types `struct ip_mreq` and `struct in_addr` are defined in `<netinet/in.h>`.

IP_TOS	(unsigned char) Sets the IP Type of Service. Allowable values for <i>optval</i> are 4 for high reliability, 8 for high throughput, and 16 for low delay. Other values will not return an error, but may have unpredictable results. Default: zero.
IP_ADD_MEMBERSHIP	(struct ip_mreq) Requests that the system join a multicast group.
IP_DROP_MEMBERSHIP	(struct ip_mreq) Allows the system to leave a multicast group.
IP_MULTICAST_IF	(struct in_addr) Specifies a network interface other than the default to be used when sending multicast datagrams through this socket. Default: multicast datagrams are sent from the interface associated with the specific multicast group, with the default multicast route or with the default route.
IP_MULTICAST_LOOP	(unsigned char ; boolean) Enables or disables loopback in the IP layer for multicast datagrams sent through this socket. The value of the variable pointed to by <i>optval</i> is zero (disable) or non-zero (enable). This option is provided for compatibility only. Normally, multicast datagrams are always looped back if the system has joined the group. See DEPENDENCIES below. Default: enabled.
IP_MULTICAST_TTL	(unsigned char) Specifies the time-to-live value for multicast datagrams sent through this socket. The value of the variable pointed to by <i>optval</i> can be zero through 255. Default: one.

IP_ADD_MEMBERSHIP requests that the system join a multicast group on the specified interface. For example:

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr = net_addr("224.1.2.3");
mreq.imr_interface.s_addr = INADDR_ANY;
setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq));
```

A system must join a group on an interface in order to receive multicast datagrams sent on the network to which that interface connects. If **imr_interface** is set to **INADDR_ANY**, the system joins the specified group on the interface that datagrams for that group would be sent from, based on the routing configuration. Otherwise, **imr_interface** should be the IP address of a local interface. An application can join up to **IP_MAX_MEMBERSHIPS** multicast groups on each socket. **IP_MAX_MEMBERSHIPS** is defined in `<netinet/in.h>`. However, each network interface may impose a smaller system-wide limit because of interface resource limitations and because the system uses some link-layer multicast addresses.

The application must also bind to the destination port number in order to receive datagrams that are sent to that port number. If the application binds to the address **INADDR_ANY**, it may receive all datagrams that are sent to the port number. If the application binds to a multicast group address, it may receive only datagrams sent to that group and port number. It is not necessary to join a multicast group in order to send datagrams to it.

IP_DROP_MEMBERSHIP allows the system to leave a multicast group. For example:

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr = net_addr("224.1.2.3");
mreq.imr_interface.s_addr = INADDR_ANY;
setsockopt(s, IPPROTO_IP, IP_DROP_MEMBERSHIP, &mreq, sizeof(mreq));
```

The system remains a member of the multicast group until the last socket that joined the group is closed or has dropped membership in the group.

IP_MULTICAST_IF specifies a local network interface to be used when sending multicast datagrams through this socket. For example:

```
#include <arpa/inet.h>
struct in_addr addr;
addr.s_addr = inet_addr("192.1.2.3");
setsockopt(s, IPPROTO_IP, IP_MULTICAST_IF, &addr, sizeof(addr));
```

Normally, applications do not need to specify the interface. By default, multicast datagrams are sent from the interface specified by the routing configuration, namely the interface associated with the specific multicast group, with the default multicast route or with the default route. If **addr** is set to the address **INADDR_ANY**, the default interface is selected. Otherwise, **addr** should be the IP address of a local interface.

IP_MULTICAST_LOOP enables or disables loopback for multicast datagrams sent through this socket. For example:

```
unsigned char loop = 1;
setsockopt(s, IPPROTO_IP, IP_MULTICAST_LOOP, &loop, sizeof(loop));
```

Note that the type of the *optval* parameter is **unsigned char** instead of **int**, which is common for boolean socket options. This option is provided for compatibility only. Normally, if a multicast datagram is sent to a group that the system has joined, a copy of the datagram is always looped back and delivered to any applications that are bound to the destination port. See **DEPENDENCIES** below.

IP_MULTICAST_TTL controls the scope a multicast by setting the time-to-live value for multicast datagrams sent through this socket. For example:

```
unsigned char ttl = 64;
setsockopt(s, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl));
```

Note that the type of *optval* parameter is **unsigned char** instead **int**, which is common for socket options. By default, the time-to-live field (TTL) is one, which limits the multicast to the local network. If the TTL is zero, the multicast is limited to the local system (loopback). If the TTL is two, the multicast can be forwarded through at most one gateway; and so forth. Multicast datagrams can be forwarded to other networks only if there are special multicast routers on the local and intermediate networks.

DEPENDENCIES

The behavior of **IP_MULTICAST_LOOP** depends on the network driver and interface card. Normally, loopback cannot be disabled, even if **IP_MULTICAST_LOOP** is set to zero, because it occurs in the driver or in the network interface. However, if the outbound interface is **lo0** (127.0.0.1), or if **IP_MULTICAST_TTL** is set to zero, setting **IP_MULTICAST_LOOP** to zero will disable loopback for multicast datagrams sent through the socket.

DIAGNOSTICS

One of the following errors may be returned if a call to **setsockopt()** or **getsockopt()** fails.

[EADDRINUSE]	The specified multicast group has been joined already on socket.
[EADDRNOTAVAIL]	The specified IP address is not a local interface address; or there is no route for the specified multicast address; or the specified multicast group has not been joined.
[EINVAL]	The parameter <i>level</i> is not IPPROTO_IP ; or <i>optval</i> is the NULL address; or the specified multicast address is not valid.
[ENOBUFS]	Insufficient memory is available for internal system data structures.
[ENOPROTOPT]	The parameter <i>optname</i> is not a valid socket option for the IPPROTO_IP level.

[EOPNOTSUPP]	The socket type is not SOCK_DGRAM .
[ETOOMANYREFS]	An attempt to join more than IP_MAX_MEMBERSHIPS multicast groups on a socket.

AUTHOR

The socket interfaces to IP were developed by the University of California, Berkeley. Multicast extensions were developed by the Stanford University.

SEE ALSO

bind(2), **getsockopt(2)**, **recv(2)**, **send(2)**, **socket(2)**, **ipMcastLoop(1M)**, **inet(7F)**.

NAME

kmem - perform I/O on kernel memory based on symbol name.

SYNOPSIS

```
#include <sys/ksym.h>
int ioctl(int kmemfd, int command, void *rks);
```

DESCRIPTION

When used with a valid file descriptor for `/dev/kmem` (*kmemfd*), `ioctl` can be used to manipulate kernel memory. The specifics of this manipulation depend on the *command* given as follows:

MIOC_READSYM

Read *mirk_buflen* bytes of kernel memory starting at the address for *mirk_symname* into *mirk_buf*. *rks* is a pointer to a `mioc_rksym` structure, defined below.

MIOC_IREADSYM

Indirect read. Read `sizeof(void *)` bytes of kernel memory starting at the address for *mirk_symname* and use that as the address from which to read *mirk_buflen* bytes of kernel memory into *mirk_buf*. *rks* is a pointer to a `mioc_rksym` structure.

MIOC_WRITESYM

Write *mirk_buflen* bytes from *mirk_buf* into kernel memory starting at the address for *mirk_symname*. *rks* is a pointer to a `mioc_rksym` structure.

MIOC_IWRITESYM

Indirect write. Read `sizeof(void *)` bytes of kernel memory starting at the address for *mirk_symname* and use that as the kernel memory address into which *mirk_buflen* bytes from *mirk_buf* are written. *rks* is a pointer to a `mioc_rksym` structure.

MIOC_LOCKSYM

Increase the hold count by one for the dynamically loaded module whose name is given by *rks*, a pointer to a character string, thereby preventing its unloading.

MIOC_UNLOCKSYM

Decrease the hold count by one for the dynamically loaded module whose name is given by *rks*, a pointer to a character string. If the count is thereby reduced to 0, the module becomes a candidate for unloading.

The `struct mioc_rksym` definition is:

```
struct mioc_rksym {
    char * mirk_modname; /* limit search for symname
                        to module modname; if NULL
                        use standard search order */
    char * mirk_symname; /* name of symbol whose address
                        is the basis for this
                        operation */
    void * mirk_buf;      /* buffer into/from which
                        read/write takes place */
    size_t mirk_buflen; /* length (in bytes) of desired
                        operation */
};
```

RETURN VALUE

`ioctl` returns 0 upon successful completion. If an error occurs, a value of -1 is returned and `errno` is set to indicate the error.

ERRORS

In addition to the errors described in *ioctl(2)*, the `kmem` `ioctl` also fails if one or more of the following are true:

- | | |
|-----------------|--|
| [EINVAL] | <i>modname</i> does not represent a currently loaded module or this is an MIOC_UNLOCKSYM and the hold count is already 0. |
| [ENXIO] | <i>kmemfd</i> open on wrong minor device (i.e., not <code>/dev/kmem</code>). |
| [EBADF] | <i>kmemfd</i> open for reading and this is an MIOC_WRITESYM . |

[ENOMATCH] *symname* not found.

[ENAMETOOLONG] *modname* is greater than **MODMAXNAMELEN** characters long, or *symname* is greater than **MAXSYMMLN** characters long.

SEE ALSO

getksym(2), ioctl(2), ioctl(5).

NAME

lan - network I/O card access information

DESCRIPTION

This manual entry gives a brief description on how to access the LAN device driver at Layer 2 (Data Link Layer) of the OSI architecture. The LAN device driver controls the various LAN interface cards (e.g. Ethernet/IEEE 802.3, FDDI, Token Ring) at Layer 1 (Physical Layer).

The Data Link Provider Interface (DLPI) is the supported method for accessing the LAN device driver at Layer 2. DLPI is intended for use by knowledgeable network users only. Refer to the *DLPI Programmer's Guide* for complete programming details.

There are HP and non-HP drivers and interface cards which will provide their own DLPI module. These types of DLPI are referred to as "native" DLPI.

Overview

The Physical Point of Attachment (PPA) is a numerical value that uniquely identifies a particular device. The PPA value can be obtained from the **lanscan** command. The "NamePPA" identifier in the **lanscan** output is a concatenation of the interface name and the PPA number. The **card instance** value for a lan device is equivalent to the PPA number for that device.

A single hardware device may have multiple "NamePPA" identifiers, which indicates multiple encapsulation methods supported for the device. For Ethernet/IEEE 802.3 links, the "Name" **lan** is used to designate Ethernet encapsulation, and **snap** for IEEE 802.3 encapsulation. For other links (FDDI, Token Ring), only the **lan** encapsulation designation is used.

Methods of transfer over the DLPI interface through the lan devices include "raw", "connectionless", and "connection-oriented" data transfers.

WARNING

The Link Level Access (LLA) interface is obsolete.

AUTHOR

lan was developed by HP.

SEE ALSO

lanscan(1M), **lanadmin(1M)**, **linkloop(1M)**.

DLPI Programmer's Guide, 1995, Hewlett-Packard

The Ethernet, A LAN: Data Link Layer and Physical Specification, Version 2.0, November 1982, Digital Equipment Corporation, Intel Corporation, Xerox Corporation

CSMA/CD Access Method and Physical Layer Specification, 1996, Institute of Electrical and Electronic Engineers

Demand-Priority Access Method, Physical Layer & Repeater Specifications, 1996, Institute of Electrical and Electronic Engineers

Fiber Distributed Data Interface (FDDI) Physical Layer Medium Dependent (PMD), 1995, ANSI

Token Ring Access Method and Physical Layer Specification, 1995, Institute of Electrical and Electronic Engineers

802.3u Media Access Control Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100 Mb/s Operation, Type 100BASE-T, 1995, Institute of Electrical and Electronic Engineers

NAME

ldterm - standard STREAMS terminal line discipline module

SYNOPSIS

```
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/termios.h>
#include <sys/bsdtty.h>
#include <sys/ttold.h>
#include <sys/strtio.h>
#include <sys/eucioctl.h>

int ioctl( fd, I_PUSH, "ldterm");
```

DESCRIPTION

ldterm is a STREAMS module that supplies the line discipline for streams-based terminal or pseudo-terminal device drivers. This module provides most of the functions of the general terminal interface described in *termio(7)*. However, it does not perform the low-level device control functions specified by the **c_cflag** word defined by the POSIX **termios** structure or the System V **termio** structure (defined in **termios.h** and **termio.h**, respectively). Also, some operations require the cooperation of the modules and drivers pushed below the **ldterm** module in a tty or pty (slave) stream. This man page only covers **ldterm** specific interface here and refers to the readers to *termio(7)* for the detail terminal interface.

Internally, the **ldterm** module uses the Extended UNIX Code (EUC) character encoding scheme. This encoding scheme enables the **ldterm** module to process multibyte characters as well as simple 8-bit characters. It correctly handles backspacing, word erasing, and tab expansion for multibyte EUC characters.

The **ldterm** module provides standard terminal operation consistent with the behavior specified by POSIX 1003.1 and System V Interface Definition (SVID) Third Edition. It also provides compatibility with the behavior of the BSD 4.3 line discipline. Notice that on other STREAMS systems, the BSD 4.3 compatibility feature is usually provided by a separate STREAMS module called **ttcompat**. Hence, applications on HP-UX need not push **ttcompat** on top of **ldterm** to get BSD 4.3 compatibility. In fact, the **ttcompat** module is not provided on the HP-UX system at all.

The **ldterm** module normally sits above either a STREAMS tty driver or a STREAMS pty slave driver. The user issues an **STREAMS I_PUSH ioctl(2)** system call to push **ldterm** onto the stream once the STREAMS tty or STREAMS pty slave device is opened.

STREAMS Messages

The **ldterm** module processes various types of STREAMS messages. The line discipline will act on any of the following message types. Any others that the module receives, however, are passed onto the next module on the stream.

Read-side Behavior

ldterm processes the following STREAMS messages on its input stream:

M_FLUSH

If **FLUSHR** is set, the read put routine flushes the read queue, discards characters in the input message buffers, and discards any partially buffered multibyte EUC characters. Then, it forwards the message upstream.

M_BREAK

The read put routine processes the message according to POSIX rules for processing **BREAK** events, parity errors, and framing errors and signal generation (see *termio(7)* for detail). If there is no data in the message, the message is assumed to represent an input **BREAK** event, which is represented by a framing error with a character value of 0 (zero). If there is data in the message, the data value is an integer that indicates the occurrence of an input **BREAK** event, or a character received with a parity or framing error. The low-order 8 bits of the data value is the byte that was read. If the **TTY_PE** flag is set in the higher-order bits of this integer, then a parity error was detected. If the **TTY_FE** flag is set in the higher-order bits of this integer, a framing error was detected.

After reading the data value, the read put routine discards the message.

M_DATA The read put routine processes the message according to the POSIX 1003.1 specification, using multibyte processing for backspacing, word erasing, and tab expansion as appropriate.

It generates echo characters and places them in the output buffer to be sent downstream to the write queue. While processing incoming data, it scans for **START** and **STOP** characters and sends **M_START**, **M_STOP** messages downstream to the write queue, if needed.

If the total number of buffered input characters is more than the high-water mark and **IXOFF** is set, the read put routine sends an **M_STOP** message downstream. When the queue reduces its backlog below the low water mark, it sends an **M_START** message downstream.

If the number of buffered input characters reaches **MAX_INPUT**, and the **IMAXBEL** flag is set, the read put routine discards new input characters and sends a **BEL** character (**Ctrl-G**) downstream. If **IMAXBEL** is not set, it flushes the input queue.

If the **ISIG** flag is set, the read put routine sends **M_PCSIG** messages upstream when the appropriate signal characters are encountered. Then it discards the characters.

If a character matching **c_cc[VDISCARD]** is encountered, and the **IEXTEN** flag is set, the read put routine sends an **M_FLUSH** (**FLUSH**) message upstream to flush all write queues. The **M_FLUSH** message is reflected by the stream head and sent downstream through all the write queues.

If the character signifies the logical termination of input, the read put routine sends the currently buffered characters upstream to the stream head.

Logical termination of input depends on the state of the **ICANON** flag. If **ICANON** is set, the **ldterm** module is in canonical input mode. In that case, the read put routine logically terminates input at the end of a line of input. Canonical line termination characters are **NEWLINE**, **EOF**, **EOL**, and **EOL2**. If **ICANON** is clear, the **ldterm** discipline module is in noncanonical or raw input mode. In that case, the read put routine terminates input when at least **VMIN** bytes are present in the input message buffer or the timer specified by **VTIME** expires (see *termio(7)* for more details).

M_IOCTL

If the message acknowledges the POSIX **termios** **TCGETS** command, the read put routine copies the **c_cflag** and speeds information, which is sent by the console driver downstream, from the message into the internal POSIX **termios** structure. Then it copies the internal POSIX **termios** structure into the message.

If the message acknowledges one of the POSIX **termios** set commands (i.e. **TCSETS**, **TCSETSW**, and **TCSETSF**) the read put routine copies all of the data from the message into the internal POSIX **termios** structure.

After this processing is done, the read put routine determines if the I/O control command was originally a BSD 4.3 or System V I/O control command that was converted to a POSIX **termios** command by the write service routine. If so, it restores the original data so that the message acknowledges the original I/O control command. Then it forwards the message upstream.

M_CTL This message was sent by the driver to make special requests to **ldterm**. The structure of **M_CTL** messages is the same as that of **M_IOCTL** messages. The **M_CTL** message block points to a message buffer containing an **iocblk** data structure (defined in *<sys/stream.h>*). The **ioc_cmd** member of this structure contains a command, just as it does in an **M_IOCTL** message. The **b_cont** member of the **M_CTL** message block contains a pointer to an **M_DATA** message block, which contains data associated with the **M_CTL** message.

The read put routine processes **M_CTL** messages containing the following commands:

MC_NO_CANON

Turn off input processing normally performed on upstream **M_DATA** messages. This is for the use of modules or drivers that perform their own input processing such as pseudo-terminal (see *ptm(7)* and *pts(7)*) in **REMOTE** mode connected to a program that performs the input processing.

MC_DO_CANON

Turn on input processing normally performed on upstream **M_DATA** messages. This message is sent when the driver want **ldterm** to exit the **REMOTE** mode.

Write-side Behavior

ldterm processes the following STREAMS messages on its output stream. Messages not listed here are simply forwarded downstream.

M_FLUSH

The write put routine flushes the write queue and discards any buffered output data. Then, it forwards the message downstream.

M_DATA The write service routine processes the data according to the POSIX 1003.1 specification output flags. It sends the processed characters downstream to the driver when the output queue fills up and all of the data is processed.

M_IOCTL

The write put routine validates the format of the **M_IOCTL** message and checks for known commands. If the message format is invalid, it turns the **M_IOCTL** message into an **M_IOCNAK** message, and returns the message upstream. If the I/O control command is not recognized, it forwards the **M_IOCTL** message downstream for processing by other modules.

The write put routine determines if the command is one that must be processed in the proper sequence relative to **M_DATA** messages. If so, it queues the **M_IOCTL** message to the write queue for later processing by the write service routine. Commands that require processing in sequence are:

TCSETSW, TCSETSF, TCSETAW, TCSETAF, TCSBRK

Otherwise, the module's write put routine processes the command immediately. Detailed descriptions of the preceding **ioctl** commands are provided in the "**ioctl Commands**" subsection, below.

M_READ This message is sent by the stream head to notify downstream modules when an application has issued a read request and there is not enough data queued at the stream head to satisfy the request. The **M_READ** is sent downstream normally when **ldterm** is operating in non-canonical input mode. If **VTIME** is positive, the write put routine starts an input timer. When the timer expires, it sends all buffered input upstream. Then, it forwards the **M_READ** message downstream.

ioctl Commands

The **ldterm** module acts on two categories of **ioctl** commands:

- Primary terminal I/O control commands
- BSD 4.3 compatibility terminal I/O control commands

Detail descriptions on how to use these **ioctls** can be found on the *termio(7)* man page. **Note:** the **FIO[xyz]** **ioctls** documented on *termio(7)* are currently not supported on **ldterm**.

Primary Terminal I/O Control Commands

The **ldterm** module acts on the following primary terminal I/O commands:

TCSETS, TCSETSW, TCSETSF

When the **ldterm** module receives any of these commands in an **M_IOCTL** message, it forwards them downstream. When it receives the **M_IOCACK** message in the read queue, it copies the POSIX **termios** information from the message into the internal POSIX **termios** structure and forwards the message upstream. If a mode change requires options at the stream head to be changed, an **M_SETOPTS** message is sent upstream. If the **ICANON** flag is turned on or off, the read mode at the stream head is changed to message-nondiscard (**RMSGN**) with read notification on (**SO_MREADON**) or byte-stream mode (**RNORM**) with read notification off (**SO_MREADOFF**), respectively. If the **TOSTOP** flag is turned on or off, the tostop mode at the stream head is turned on (**SO_TOSTOP**) or off (**SO_TONSTOP**), respectively.

TCGETS The **ldterm** module forwards the **M_IOCTL** message downstream. When it receives the **M_IOCACK** message in the read queue, it copies the **CLOCAL** flags and speeds from the message into the internal POSIX **termios** structure. Then, it copies the entire structure into the **M_IOCACK** message and forwards the message upstream.

TCSETA, TCSETAW, TCSETAF

These commands set the old System V **termio** information. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL** message, then forwards the message with a corresponding POSIX **termios** command (i.e. **TCSETS, TCSETSW, TCSETSF**). The original

I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TCGETA This command get the old System V **termio** information. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL** message, then forwards the message with the **TCGETS** command. The original I/O control command and **M_IOCTL** message are stored to be used on **M_IOCACK**. When it receives the matching **M_IOCACK** message, the **ldterm** module processes it as for a **TCGETS** command, then converts the POSIX **termios** information into the System V **termio** information and replies.

TCSBRK The **ldterm** module forwards this command downstream to be handled by the driver so that the driver has a chance to drain the data before sending an **M_IOCACK** message upstream.

TCXONC This command controls the behavior of input/output flow control. If the argument is 0 and output is not already stopped, an **M_STOP** message is sent downstream. If the argument is 1 and the output is stopped, an **M_START** message is sent downstream. If the argument is 2 and input is not already stopped, an **M_STOPI** message is sent downstream. If the argument is 3 and input is stopped, an **M_STARTI** message is sent downstream.

TCFLSH This command flush the input or/and output streams. If the argument is 0, an **M_FLUSH** message with a flag byte of **FLUSHR** is sent downstream. This **M_FLUSH(FLUSHR)** message will be reflected back upstream by the driver to flush the entire input stream. If the argument is 1, an **M_FLUSH** message with a flag byte of **FLUSHW** is sent upstream. This **M_FLUSH(FLUSHW)** message will be reflected downstream by the stream head to flush the entire output stream.

TIOCSWINSZ

This command sets the window size variables. The argument of this command takes a pointer to a **winsize** structure. The **ldterm** module does not use the window size variable, but maintains it here for any needed replies to **TIOCGWINSZ** commands. The module forwards the message downstream.

TIOCGWINSZ

When the **ldterm** module receives this command, it returns the window size variable that was set by the last **TIOCSWINSZ** command. The argument of this command takes a pointer to a **winsize** structure.

EUC_WSET

This command sets the character widths and screen widths for the EUC character sets. The argument of this command takes a pointer to an **eucluc** structure which contains the information for setting the character widths and screen widths of the EUC character sets. After processing the command, **ldterm** forwards this message downstream to the next module.

EUC_WGET

This command returns the character widths and screen widths for the EUC character sets. This command takes a pointer to an **eucluc** structure via which the EUC character widths and screen widths information will be returned.

EUC_SET_HP15

This command put **ldterm** to the so called **HP15** mode which enable **ldterm** to recognize the **HP15_SJIS**, **HP15_BIG5**, **HP15_CCDC**, and **HP15_GB** character sets and process them in such a way that they behave like EUC characters. The argument for this command takes a pointer to an integer value which specify on of the above-mentioned four supported HP15 character sets. If the argument is set to **HP15_ASCII**, then **ldterm** will switch back to normal ASCII processing. **EUC_WSET** is mutually exclusive with **EUC_SET_HP15**.

EUC_GET_HP15

This command returns the current HP15 character that has been set via the **EUC_SET_HP15** command. This command takes a pointer to an integer via which the result is returned. If no previous **EUC_SET_HP15** has been issued, then it will return **HP15_ASCII**.

BSD 4.3 Compatible Terminal I/O Commands

The **ldterm** module acts on the following I/O commands, which are compatible with the BSD I/O environment:

TIOCEXCL

Set 'exclusive-use' mode. No further opens are permitted until the file has been closed.

TIOCNXCL

Turn off 'exclusive-use' mode.

TIOCSETD

The **ldterm** module does nothing but reply to this command. In a BSD system, the command is used to set the current line discipline type. It does not have much meaning in a STREAMS environment, because line discipline modules are changed by popping the current module from the stream and pushing a different one onto the stream.

TIOCGETD

In a BSD system, this command is used to get the current line discipline type. The command does not have much meaning in a STREAMS environment. The **ldterm** module replies with a value of 2 for binary compatibility, since **ldterm** supports job control.

TIOCFUSH

This command flush the input or/and output streams similar to that of the **TCFLSH** command. The argument is a pointer to an **int** variable. If its value is zero, both the input and output streams are flushed by sending the appropriate **FLUSHR/FLUSHW M_FLUSH** messages upstream and downstream. Otherwise, the value of the **int** is treated as the logical **OR** of the **FREAD** and **FWRITE** flags defined by **<sys/file.h>**. If the **FREAD** flag is set, the input stream is flushed. If the **FWRITE** flag is set, the output stream is flushed. Then, **ldterm** acknowledges the message with **M_IOCACK**.

TIOCOUTQ

This command takes a pointer to an integer and returns the number of characters buffered up in the **ldterm's** output buffer.

TIOCHPCL

This command sets the POSIX **termios HUPCL** flag to indicate that the terminal line should be disconnected when the last file descriptor associated with that line is closed. The **ldterm** module converts the command into a compatible POSIX **termios** I/O control command by sending an **M_IOCTL** message containing the **TCSETS** command with current **termios** settings downstream.

TIOCSTART

The command restarts output. If the terminal was stopped, the **ldterm** module sends an **M_START** message downstream.

TIOCSTOP

This command stops output. The **ldterm** module sends an **M_STOP** message downstream.

TIOCSBRK

This command sets the break condition on a line. The **ldterm** module sends an **M_BREAK** message containing a value of 1 as data to the driver, then replies with **M_IOCACK**.

TIOCCBRK

This command clears the break condition on a line. The **ldterm** module sends an **M_BREAK** message containing a value of 0 (zero) as data to the driver, then replies with **M_IOCACK**.

TIOCSETP, TIOCSETN

These commands set the **sgttyb** information, defined in **<sys/ttold.h>**. The argument is a pointer to an **sgttyb** structure. The **ldterm** module converts the message to a POSIX **termios M_IOCTL** message. Then, it forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETSW, TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGETP

This command returns the **sgttyb** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**. The argument is a pointer to an **sgttyb** structure via where the information is returned.

TIOCSETC

This command sets the **tchars** information, defined in **<sys/strtio.h>**. The argument is a pointer to an **tchars** structure. The **ldterm** module converts the message to a POSIX **termios M_IOCTL** message. Then, it forwards the POSIX **termios M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGGETC

This command returns the **tchars** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**. The argument is a pointer to an **tchars** structure via where the information is returned.

TIOCSLTC

This command sets the **ltchars** information defined in `<sys/bsdttty.h>`. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL** message. Then, it forwards the POSIX **termios** **M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). The original I/O control command and **M_IOCTL** message are stored for use on **M_IOCACK**.

TIOCGLTC

The **ldterm** module returns the **ltchars** information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**.

TIOCLBIS, TIOCLBIC, TIOCLSET

These commands set the BSD 4.3 flags information, defined in `<sys/strtio.h>`. For **TIOCLBIS** and **TIOCLBIC**, the argument is a pointer to an **int** whose value is a mask containing flags to be set/clear. For **TIOCLSET**, the argument is a pointer to an **int** whose value is a new set of flags to be set. The **ldterm** module converts the message to a POSIX **termios** **M_IOCTL**, then forwards the POSIX **termios** **M_IOCTL** message with a corresponding POSIX **termios** command (i.e. **TCSETS**). It stores the original I/O control command and **M_IOCTL** message to be used on **M_IOCACK**.

TIOCLGET

The **ldterm** module returns the BSD 4.3 flags information based on the interpretation of the current content of the POSIX **termios** structure maintained in **ldterm**.

TIOCS TI

This command takes an argument of a pointer to a character and pretends that the character was typed on the terminal. The user must either have superuser privilege or have read permission on the controlling terminal which the **ioctl** is issued against.

FIONREAD

This command takes an argument of a pointer to an integer and returns the number of immediately readable characters.

AUTHOR

ldterm was developed by HP and OSF.

SEE ALSO

ioctl(2), **streamio**(7), **termio**(7), **ptm**(7), **pts**(7), **ptem**(7).

NAME

lp - line printer

SYNOPSIS

```
#include <sys/lprio.h>
```

Remarks:

This manual entry applies only to a certain group of printers. For Series 800 systems, it applies to printers controlled by the device drivers `lpr0`, `lpr1`, and `lpr2`. It does *not* apply to any printers on Series 700 systems.

DESCRIPTION

This section describes capabilities provided by many line printers supported by various versions of the HP-UX operating system. A line printer is a character special device that may optionally have an interpretation applied to the data.

If the character special device file has been created with the raw option (see the HP-UX System Administrator manuals for information about creating device files with the raw option), data is sent to the printer in **raw mode** (as, for example, when handling a graphics printing operation). In raw mode, no interpretation is done on the data to be printed, and no page formatting is performed. Data bytes are simply sent to the printer and printed exactly as received.

If the device file does not contain the raw option, data can still be sent to the printer in raw mode. Raw mode is set and cleared by the LPRSET request.

If the line printer device file does not contain the raw option, data is interpreted according to rules discussed below. The driver understands the concept of a printer page in that it has a page length (in lines), line length (in characters), and offset from the left margin (in characters). The default line length, indent, lines per page, open and close page eject, and handling of backspace are set to defaults determined when the printer is opened and recognized by the system the first time. If the printer is not recognized, the default line length is 132 characters, indent is 4 characters, lines per page is 66, one page is ejected on close and none on open, and backspace is handled for a character printer.

The following rules describe the interpretation of the data stream:

- A form feed causes a page eject and resets the line counter to zero.
- Multiple consecutive form-feeds are treated as a single form-feed.
- The new-line character is mapped into a carriage-return/line-feed sequence, and if an offset is specified a number of blanks are inserted after the carriage-return/line-feed sequence.
- A new-line that extends over the end of a page is turned into a form-feed.
- Tab characters are expanded into the appropriate number of blanks (tab stops are assumed to occur every eight character positions as offset by the current indent value).
- Backspaces are interpreted to yield the appropriate overstrike either for a character printer or a line printer.
- Lines longer than the line length minus the indent (i.e., 128 characters, using the above defaults) are truncated.
- Carriage-return characters cause the line to be overstruck.
- When it is opened or closed, a suitable number of page ejects is generated.

Two `ioctl(2)` requests are available to control the lines per page, characters per line, indent, handling of backspaces, and number of pages to be ejected at open and close times. At either open or close time, if no page eject is requested the paper will not be moved. For opens, line and page counting will start assuming a top-of-form condition.

The `ioctl` requests have the following form:

```
#include <sys/lprio.h>

int ioctl(int fildes, int request, struct lprio *arg);
```

The possible values of *request* are:

LPRGET Get the current printer status information and store in the `lprio` structure to which *arg* points.

(Series 800 Only)

LPRSET Set the current printer status information from the structure to which *arg* points.

The **lprio** structure used in the LPRGET and LPRSET requests is defined in **<sys/lprio.h>**, and includes the following members:

```
short int  ind;          /* indent */
short int  col;          /* columns per page */
short int  line;         /* lines per page */
short int  bksp;         /* backspace handling flag */
short int  open_ej;      /* pages to eject on open */
short int  close_ej;     /* pages to eject on close */
short int  raw_mode;     /* raw mode flag */
```

These are remembered across opens, so the indent, page width, and page length can be set with an external program. If the **col** field is set to zero, the defaults are restored at the next open.

If the backspace handling flag is 0, a character printer is assumed and backspaces are passed through the driver unchanged. If the flag is a 1, a line printer is assumed, and sufficient print operations are generated to generate the appropriate overstruck characters.

If the raw mode flag is 0, data sent to the printer is formatted according to indent, columns per page, lines per page, backspace handling, and pages to eject on open and close.

If the raw mode flag is 1, data sent to the printer is not formatted.

If the raw mode flag is changed from 1 to 0 (raw mode is turned off) and the format settings (indent, columns per page, etc.) have not been modified, the data is formatted according to the prior format settings.

AUTHOR

lp was developed by HP and AT&T.

FILES

```
/dev/lp          default or standard printer used by some HP-UX commands;
/dev/[r]lp*      special files for printers
```

SEE ALSO

lp(1), slp(1), loctl(2), cent(7), intro(7).

NAME

lvm - Logical Volume Manager (LVM)

DESCRIPTION

The Logical Volume Manager (LVM) is a subsystem for managing disk space. The HP LVM subsystem offers value-added features, such as mirroring (with the optional HP MirrorDisk/UX software), high availability (with the optional HP ServiceGuard software), and striping, that enhance availability and performance.

Unlike earlier arrangements where disks were divided into fixed-sized sections, LVM allows the user to consider the disks, also known as **physical volumes**, as a pool (or volume) of data storage, consisting of equal-sized extents. The default size of an extent is 4 MB.

An LVM system consists of arbitrary groupings of physical volumes, organized into **volume groups**. A volume group can consist of one or more physical volumes. There can be more than one volume group in the system. Once created, the volume group, and not the disk, is the basic unit of data storage. Thus, whereas earlier one would move disks from one system to another, with LVM, one would move a volume group from one system to another. For this reason it is often convenient to have multiple volume groups on a system.

Volume groups can be subdivided into virtual disks, called **logical volumes**. A logical volume can span a number of physical volumes or represent only a portion of one physical volume. The pool of disk space that is represented by a volume group can be apportioned into logical volumes of various sizes. The size of a logical volume is determined by its number of extents. Once created, logical volumes can be treated just like disk partitions. Logical volumes can be assigned to file systems, used as swap or dump devices, or used for raw access.

Commands

LVM information can be created, displayed, and manipulated with the following commands:

lvchange	Change logical volume characteristics
lvcreate	Stripe, create logical volume in volume group
lvdisplay	Display information about logical volumes
lvextend	Increase space, increase mirrors for logical volume
lvlnboot	Prepare logical volume to be root, primary swap, or dump volume
lvmmigrate	Prepare root file system for migration from partitions to logical volumes
lvreduce	Decrease number of physical extents allocated to logical volume
lvremove	Remove one or more logical volumes from volume group
lvrmboot	Remove logical volume link to root, primary swap, or dump volume
pvchange	Change characteristics of physical volume in volume group
pvcreate	Create physical volume for use in volume group
pvddisplay	Display information about physical volumes within volume group
pvmmove	Move allocated physical extents from one physical volume to other physical volumes
vgcfgbackup	Create or update volume group configuration backup file
vgcfgrestore	Display or restore volume group configuration from backup file
vgchange	Set volume group availability
vgcreate	Create volume group
vgdisplay	Display information about volume groups
vgexport	Export an volume group and its associated logical volumes
vgextend	Extend an volume group by adding physical volumes
vgimport	Import an volume group onto the system
vgreduce	Remove physical volumes from an volume group
vgremove	Remove volume group definition from the system
vgscan	Scan physical volumes for volume groups

The following commands are also available if the HP MirrorDisk/UX software is installed:

lvmerge	Merge two logical volumes into one logical volume
lvsplit	Split mirrored logical volume into two logical volumes
lvsync	Synchronize stale mirrors in logical volumes
vgsync	Synchronize stale logical volume mirrors in volume groups

EXAMPLES

The basic steps to take to begin using LVM are as follows:

- Identify the disks to be used for LVM.
- Create an LVM data structure on each identified disk (see *pvccreate*(1M)).
- Collect all the physical volumes to form a new volume group (see *vgcreate*(1M)).
- Create logical volumes from the space in the volume group (see *lvcreate*(1M)).
- Use each logical volume as if it were a disk section (create a file system, or use for raw access).

To configure disk `/dev/dsk/c0t0d0` as part of a new volume group named **vg01**.

First, initialize the disk for LVM with the **pvccreate** command.

```
pvccreate /dev/rdisk/c0t0d0
```

Then, create the pseudo device file that is used by the LVM subsystem.

```
mkdir /dev/vg01  
mknod /dev/vg01/group c 64 0x030000
```

The minor number for the **group** file should be unique among all the volume groups on the system. It has the format `0xNN0000`, where *NN* ranges from 00 to 09. The maximum value of *NN* is controlled by the kernel tunable parameter, **maxvgs**.

Create the volume group, **vg01**, containing the physical volume, `/dev/dsk/c0t0d0`, with the **vgcreate** command.

```
vgcreate /dev/vg01 /dev/dsk/c0t0d0
```

You can view information about the newly created volume group with the **vgdisplay** command.

```
vgdisplay -v /dev/vg01
```

Create a logical volume of size 100 MB, named **usrvol**, on this volume group with the **lvcreate** command.

```
lvcreate -L 100 -n usrvol /dev/vg01
```

This creates two device files for the logical volume, `/dev/vg01/usrvol`, which is the block device file, and `/dev/vg01/rusrvol`, which is the character (raw) device file.

You can view information about the newly created logical volume with the **lvdisplay** command.

```
lvdisplay /dev/vg01/lvol1
```

Any operation allowed on a disk partition is allowed on the logical volume. Thus, you can use **usrvol** to hold a file system.

```
newfs /dev/vg01/rusrvol hp7937  
mount /dev/vg01/usrvol /usr
```

SEE ALSO

lvchange(1M), *lvcreate*(1M), *lvdisplay*(1M), *lvextend*(1M), *lvlnboot*(1M), *lvreduce*(1M), *lvremove*(1M), *lvrmboot*(1M), *pvchange*(1M), *pvccreate*(1M), *pvddisplay*(1M), *pvmmove*(1M), *vgcfgbackup*(1M), *vgcfgrestore*(1M), *vgchange*(1M), *vgcreate*(1M), *vgdisplay*(1M), *vgexport*(1M), *vgextend*(1M), *vgimport*(1M), *vgreduce*(1M), *vgremove*(1M), *vgscan*(1M).

Managing Systems and Workgroups.

If HP MirrorDisk/UX is installed: *lvmerge*(1M), *lvsplit*(1M), *lvsync*(1M), *vgsync*(1M).

If HP ServiceGuard is installed: *cmcheckconf*(1M), *cmquerycl*(1M), *Managing MC/ServiceGuard*.

NAME

mem - main memory

DESCRIPTION

mem is a special file that is an image of the main memory of the computer. It may be used, for example, to examine and patch the system.

Byte addresses in *mem* are interpreted as physical memory addresses. References to non-existent locations cause errors to be returned.

File *kmem* is the same as *mem* except that kernel virtual memory rather than physical memory is accessed. Please refer to *kmem(7)* for information about ioctl operations that are supported on **/dev/kmem**.

WARNINGS

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

FILES

/dev/mem
/dev/kmem

SEE ALSO

kmem(7)

NAME

modem - asynchronous serial modem line control

SYNOPSIS

```
#include <sys/modem.h>
```

DESCRIPTION

This section describes the two modes of modem line control and the three types of terminal port access. It also discusses the effect of the bits of the *termio* structure that affect modem line control. The modem related *ioctl(2)* system calls are discussed at the end of the document.

Definitions

There are several terms that are used within the following discussion which will be defined here for reference. "Modem control lines" (CONTROL) are generally defined as those outgoing modem lines that are automatically controlled by the driver. "Modem status lines" (STATUS) are generally defined as those incoming modem lines that are automatically monitored by the driver. CONTROL and STATUS for a terminal file vary according to the modem line control mode of the file (see *Modem line control modes* below). An *open(2)* to a port will be considered to be BLOCKED if it is waiting for another file on the same port to be closed. An *open* to a port is considered to be PENDING if it is waiting for the STATUS to be raised. An *open* to a port is considered to be SUCCESSFUL if the *open* system call has returned to the calling process without error.

Open flag bits

Currently, the only *open* flag bits recognized by the driver is the O_NDELAY and O_NONBLOCK bits. When either of these bits is set, an *open* call to the driver will never become blocked. If possible, the *open* will be returned immediately as SUCCESSFUL, and the driver will continue the process of opening the tty file. If it is not possible, then the *open* will be returned immediately with the appropriate error code as described in the appropriate section.

Termio bits

When set, the CLOCAL bit in the *termios* or *termio* structure (see *termio(7)*) is used to remove the driver's automatic monitoring of the modem lines. However, the user's ability to control the modem lines is determined only by the mode in effect and does not depend on the state of CLOCAL. Normally, the driver will monitor and require the STATUS to be raised. An *open* system call will raise the CONTROL and wait for the STATUS before completing unless the CLOCAL bit is set. (If the O_NDELAY or O_NONBLOCK bit is set, the *open* will be returned immediately, but the driver will otherwise continue to monitor the modem lines as normal based on the state of the CLOCAL bit.) Normally, loss of the STATUS will cause the driver to break the modem connection and lower the CONTROL. However, if CLOCAL is set, any changes in the STATUS will be ignored. A connection is required before any data may be read or written, unless CLOCAL is set. Any timers that would normally be in effect (see *Modem line control modes* and *Modem timers* below) will be stopped while CLOCAL is set.

When the CLOCAL bit is changed from clear to set, the driver will assume the existence of an active device (such as a modem) on the port regardless of the STATUS. If any of the CONTROL are raised at that point in time, they will continue in that state. The STATUS will no longer be actively monitored. When the CLOCAL bit is changed from set to clear, the driver will resume actively monitoring the STATUS. If all of the CONTROL and STATUS are raised at that point in time, the driver will continue the modem connection. If any of the STATUS are not raised, the driver will act as though those signals were lost (as described in *Modem line control modes* below) and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process. If any of the CONTROL are not raised, the driver will break the modem connection by lowering all the CONTROL.

The HUPCL bit in the *termios* or *termio* structure determines the action of the driver regarding the CONTROL when the last *close* system call is issued to a terminal file. If the HUPCL bit is set, the driver will lower the CONTROL at *close* time and the modem connection will be broken. If HUPCL is not set and a modem connection exists, it will continue to exist, even after the *close* is issued. The driver will not change the CONTROL.

Terminal port access types

There are three types of modem access: call-in connections, call-out connections, and direct (no modem control) connections. A given port may be accessed through all three types of connection by accessing different files. The modem access type of a terminal file is determined by the file's major and/or minor device numbers.

The call-in type of access is used when the connection is expected to be established by an incoming call. This is the type that would be used by *getty*(1M) to accept logins over a modem. When an *open* is issued to such a file, the driver may wait for an incoming call and will then raise the CONTROL based on the current mode (see below) of the port. When the port is closed, the driver may or may not lower the CONTROL depending on the HUPCL bit.

The call-out type of access is used when the connection is expected to be established by an outgoing call. This would be used by programs such as *uucp*(1). When an *open* is issued to such a file, the driver will immediately raise the CONTROL and wait for a connection based on the mode currently in effect. When the port is closed, the driver may or may not lower the CONTROL depending on the HUPCL bit.

The direct type of access is used when no driver modem control is desired. This could then be used for directly connected terminals that use a three-wire connection, or to talk to a modem before a connection has been established. The second case allows a program to give dialing instructions to the modem. Neither the CLOCAL nor the HUPCL bits have any effect on a port accessed through a direct file. (However, both bits may be inherited by other types of files; see *Terminal port access interlock* below.) An *open* to a direct file does not affect the CONTROL and does not depend on any particular state of the STATUS to succeed. When the file is closed, the driver will not affect the state of the CONTROL. If a modem connection has been established, it will continue to exist. Setting the speed of a direct file to B0 (see *termio*(7)) will be considered an impossible speed change and will be ignored. It will not affect the CONTROL.

Modem line control modes

There are two modes of modem line control: CCITT mode and simple mode. A given port may have only one of these two modes in effect at any given point in time. An attempt to *open* a port with a mode other than the one in effect (from a PENDING or SUCCESSFUL *open* on a different file) will cause the *open* to be returned with an ENXIO error. The modem access type of a terminal file is determined by the file's major and/or minor device numbers.

CCITT mode is used for connections to switched line modems. The CONTROL for CCITT mode are Data Terminal Ready (DTR) and Request to Send (RTS). The STATUS are Data Set Ready (DSR), Data Carrier Detect (DCD), and Clear to Send (CTS). Additionally, the Ring Indicator (RI) signal indicates the presence of an incoming call. When a connection is begun (an incoming call for a call-in file or an *open* issued to a call-out file), the CONTROL are raised and a connection timer (see *Modem timers* below) is started. If the STATUS become raised before the time period has elapsed, a connection is established and the *open* request is returned successfully. If the time period expires, the CONTROL are lowered and the connection is aborted. For a call-in file, the driver will wait for another incoming call; for a call-out file, the *open* will be returned with an EIO error. Once a connection is established, loss of either DSR or CTS will cause the CONTROL to be lowered and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process.

If DCD is lost, a timer is started. If DCD resumes before the time period has expired, the connection will be maintained. However, no data transfer will occur during this time. The driver will stop transmitting characters, and any characters received by the driver will be discarded. (However, on some implementations data transmission cannot be stopped. See *DEPENDENCIES*.) If DCD is not restored within the allotted time, the connection will be broken as described above for DSR and CTS.

If the modem connection is to be broken when the *close* system call is issued (i.e. HUPCL is set), then the CONTROL will be lowered and the *close* will be returned as successful. However, no further *opens* will be allowed until after both DSR and CTS have been lowered by the modem, and the hangup timer (see *Modem timers* below) has expired. The action taken in response to an *open* during this time will be the same as if the port were still open. (See *Terminal port access interlock* below.)

When a port is in CCITT mode, the driver has complete control of the modem lines and the user is not allowed to change the setting of the CONTROL or affect which STATUS are actively monitored by the driver (see *Modem ioctls* below). This is to provide strict adherence with the CCITT recommendations.

Simple mode is used for connections to devices which require only a simple method of modem line control. This can include devices such as black boxes, data switches, or for system-to-system connections. It can also be used with modems which cannot operate under the CCITT recommendations. The CONTROL for simple mode consists of only DTR. The STATUS consists of only DCD. When an *open* is issued, the CONTROL is raised but no connection timer is started. When the STATUS becomes raised, a connection is established and the *open* request is returned as SUCCESSFUL. Once a connection is established, loss of the STATUS will cause the CONTROL to be lowered and, if the device is a controlling terminal, a *hangup* signal will be sent to the controlling process.

When a port is in simple mode, the driver will normally control the modem lines. However, the user is allowed to change the setting of the CONTROL (see *Modem ioctls* below).

Terminal port access interlock

An interlock mechanism is provided between the three access types of terminal files. It prevents more than one file from being successfully opened at a time, but allows certain *opens* to succeed while others are PENDING so that a port can be opened through a call-out connection while *getty* has a pending *open* at a call-in connection. The three access types are given a priority that determines which *open* will succeed if more than one file has an *open* issued against it. The three access types are ordered from lowest priority to highest as follows: call-in, call-out, and direct.

If an *open* is issued to a port which already has a SUCCESSFUL *open* on it of a lower priority type, the new *open* will be returned with an EBUSY error. (EBUSY will also be returned by an attempted *open* on a CCITT call-out file if an incoming call indication is currently being received. In this case, if there is a PENDING *open* on the corresponding CCITT call-in file, this PENDING *open* will complete.) If the lower priority *open* is PENDING, the new *open* will succeed if possible, or will be left PENDING if waiting for the STATUS and the lower priority *open* will become BLOCKED. If a higher priority *open* has succeeded or is PENDING, the new *open* will be BLOCKED, unless the new *open* has the O_NDELAY flag bit set, in which case the *open* will be returned with an EBUSY error. Once an *open* on one type of file is SUCCESSFUL, any PENDING *opens* on lower priority files will become BLOCKED.

When a file of one priority is closed, a BLOCKED *open* on the next lower priority type file will become active. If all of the STATUS are raised, the *open* will be SUCCESSFUL, otherwise the *open* will become PENDING waiting for the STATUS. If the lower priority *open* is SUCCESSFUL (because the connection was maintained when the higher priority file was closed), the port characteristics (speed, parity, etc.) that were set by the higher priority file will be inherited by the lower priority file. If the connection is not maintained through the *close*, the port characteristics will be set to default values.

Modem timers

There are four timers currently defined for use with modem connections. The first three of the timers are applicable only to CCITT mode connections. In general, the effect of changing a timer value while the timer is running is system dependent. However, setting the timer value to zero is guaranteed to disable the timer even if it is running.

The connect timer is used to limit the amount of time to wait for a connection to be established once it has been begun. This timer is started when an incoming call has been received on a call-in file, or when an *open* has been issued on a call-out file for which no *opens* are already pending. If the connection is completed in time, the timer is aborted. If the time period expires, the connection is aborted. For a call-in file, the driver will again wait for an incoming call and the *open* will remain pending. For a call-out file, the *open* will be returned with an EIO error.

The carrier detect timer is used to limit the amount of time to wait before causing a disconnect if DCD drops. If carrier is not re-established in this time, a disconnect will occur. If carrier is re-established before the timeout, the timer will be aborted and the connection maintained. During the period when carrier is not raised, no data will be transferred across the line.

The no activity timer is used to limit the amount of time a connection will remain open with no data transfer across the line. When the data line becomes quiescent with no data transfer, this timer will be started. If data is again transferred over the line in either direction before the time limit, the timer will be aborted. If no activity occurs before the timeout has occurred, the driver will disconnect the line. This can be used to avoid long and costly telephone connections when data transfer has been stopped either normally or abnormally.

The last timer defined, the hangup timer, is used for both CCITT and simple modes. This timer controls the amount of time to wait after disconnecting a modem line before allowing another *open*. This time period should be made long enough to guarantee that the connection has been terminated by the telephone switching equipment. If this period is not long enough, the telephone connection may not be broken and a succeeding *open* may complete with the old connection.

HP-UX Modem ioctl

Several *ioctl* system calls apply to manipulation of modem lines. They use the following information defined in `<sys/modem.h>`:

```
#define NMTIMER 6
typedef unsigned long mflag;
struct mtimer {
    unsigned short m_timers[NMTIMER];
};
```

Each bit of the *mflag* long corresponds to one of the modem lines as follows:

MRTS	Request to Send	outbound
MCTS	Clear to Send	inbound
MDSR	Data Set Ready	inbound
MDCD	Data Carrier Detect	inbound
MDTR	Data Terminal Ready	outbound
MRI	Ring Indicator	inbound
MDRS	Data Rate Select	outbound

The timer values are defined in the array `m_timers`. The relative position of the timer and default initial values and units for each timer are as follows:

0	MTCONNECT	25 s
1	MTCARRIER	400 ms
2	MTNOACTIVITY	0 min
3	MTHANGUP	250 ms
4	Reserved	
5	Reserved	

A value of zero for any timer will disable that timer.

The modem line *ioctl* system calls have the form:

```
int ioctl(int fildes, int command, mflag *arg);
```

The commands using this form are:

MCGETA	Get the current state of both inbound and outbound modem lines and store in the <i>mflag</i> long referenced by <i>arg</i> . A raised line will be indicated by a one bit in the appropriate position.
MCSETA	Set the outbound modem lines from the <i>mflag</i> long referenced by <i>arg</i> . Setting an outbound bit to one causes that line to be raised and zero to be lowered. Setting bits for inbound lines has no effect. Setting any bits while in CCITT mode has no effect. The change to the modem lines is immediate and using this form while characters are still being output may cause unpredictable results.
MCSETAW	Wait for the output to drain and set the new parameters as described above.
MCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters as described above.

The timer value *ioctl* system calls have the form:

```
int ioctl(int fildes, int command, mtimer *arg);
```

The commands using this form are:

MCGETT	Get the current timer value settings and store in the <i>mtimer</i> structure referenced by <i>arg</i> .
MCSETT	Set the timer values from the structure referenced by <i>arg</i> .

For any timer, setting the timer value to its previous value has no effect.

SVID3 Modem ioctls

System V Interface Definition, Third Edition (SVID3) specifies additional *ioctl* system calls to manipulate the modem lines. They use information defined in `<termios.h>`.

Each *ioctl* passes an integer argument in which each of the following bit definitions correspond to one of the modem lines as follows:

TIOCM_RTS	Request to Send	outbound
TIOCM_CTS	Clear to Send	inbound
TIOCM_DSR	Data Set Ready	inbound
TIOCM_CAR	Data Carrier Detect	inbound
TIOCM_DTR	Data Terminal Ready	outbound
TIOCM_RNG	Ring Indicator	inbound

Additionally, `TIOCM_CD` is equivalent to `TIOCM_CAR`, and `TIOCM_RI` is equivalent to `TIOCM_RNG`.

The modem line *ioctl* system calls have the form:

```
int ioctl(int fildes, int command, int *arg);
```

The commands using this form are:

TIOCMGET	Get the current state of both inbound and outbound modem lines and store in the int referenced by arg . A raised line will be indicated by a one bit in the appropriate position.
TIOCMSET	Set the outbound modem lines from the int referenced by arg .
TIOCMBIS	Raise the control lines specified by a one in the corresponding bit positions of the int referenced by arg .
TIOCMBIC	Lower the control lines specified by a one in the corresponding bit positions of the int referenced by arg .

Note that setting bits for inbound lines has no effect, and setting any bits while in CCITT mode has no effect. Also, the change to the modem lines is immediate and using these ioctl's while characters are still being output may cause unpredictable results.

WARNING

Occasionally it is possible that a process may open a call-out file at approximately the same time as an incoming call is received. In some cases, the call-out connection may be satisfied by the incoming call. In general, however, the results are indeterminate. If necessary, the situation can be avoided by the use of two modems and ports, one for call-out connections and the other for receiving incoming calls.

DEPENDENCIES

Some hardware implementations may not have access to all modem lines supported by MCSETA. If a particular hardware does not support a given line, attempts to set the value of a line will be ignored, and reading the current state of the line will return zero. The appropriate I/O card manual should be referenced to determine the lines supported by the hardware installed.

Some hardware implementations may not have access to all timers supported by MCSETT. Also, the granularity of the individual timers may vary depending on the hardware and system in use. The effect of setting a timer out of range or with a granularity outside the capability of a particular system should be documented by that system. The effect of changing the value for a timer while that timer is running is system dependent and should be documented by each system.

Setting the CLOCAL bit while a timer is running will cause the timer to be stopped. It is a system dependency whether or not the timer is restarted, and if so, the value at which it is restarted when the CLOCAL bit is subsequently cleared.

On those implementations supporting the HP27140A 6-Channel Multiplexer, transmission of characters cannot be stopped during loss of DCD. The driver cannot detect loss of DCD until the connection is broken. Also, the I/O card may still have characters in its internal buffers and will still try to transmit them.

AUTHOR

modem was developed by HP and AT&T.

FILES

```
/dev/cua *
/dev/cul *
/dev/tty *
/dev/ttyd *
```

EISA mux has adopted the following device files naming convention:

```
/dev/<cua | cul | tty | ttyd><Instance_Number><Port_Module_Name><Port_Number>
```

where *Port_Module_Name* = [a..h], *Port_Number* = 1-16

For example: */dev/cua2a1*, */dev/cul2b8*, */dev/tty2g12*, */dev/ttyd2h16*

SEE ALSO

stty(1), mknod(1M), ioctl(2), open(2), termio(7).

NAME

mt - magnetic tape interface and controls for stape, tape1 and tape2

DESCRIPTION

This entry describes the behavior of HP magnetic tape interfaces and controls, including reel-to-reel, DDS, QIC, 8mm, and 3480 tape drives. The files `/dev/rmt/*` refer to specific raw tape drives, and the behavior of each given unit is specified in the major and minor numbers of the device special file.

Naming Conventions

There are two naming conventions for device special files. The standard (preferred) convention is used on systems that support long file names. An alternate convention is provided for systems limited to short file names. The following standard convention is recommended because it allows for all possible configuration options in the device name and is used by *mksf(1M)* and *insf(1M)*:

```
/dev/rmt/c #t #d #o [[z][e][p][s][#]][w]density[C[#]][n][b]
```

The following alternate naming convention is provided to support systems in which the `/dev/rmt` directory requires short file names. These device special file names are less descriptive, but guarantee unique device naming and are used by *mksf(1M)* and *insf(1M)* where required.

```
/dev/rmt/c #t #d #f #| i #|n][b]
```

For each tape device present, eight device files are automatically created when the system is initialized. Four of these device files utilize either the standard (long file name) or alternate (short file name) naming conventions. When the standard naming convention is being utilized, these four files contain the density specification "BEST". When the alternate naming convention is being utilized, these four files contain the density specification "f0". There are four such files because each of the four different permutations of the "n" and "b" options (see below) is available.

The remaining four files automatically created when the system is initialized utilize the pre-HP-UX 10.0 device file naming convention. This includes an arbitrary number to distinguish this tape device from others in the system, followed by the letter **m**. There are four such files because each of the four different permutations of the **n** and **b** options (see below) is available. These files are created as a usability feature for pre-HP-UX 10.0 users who do not wish to acquire familiarity with the standard or alternate naming conventions.

Each of the automatically created four device files which utilize the standard or alternate naming conventions is linked to a device file which utilizes the pre-HP-UX 10.0 naming convention. Thus, the device files which utilize the pre-HP-UX 10.0 naming convention provide the same functionality as the device files which contain the density specification **BEST** (standard naming convention) or **f0** (alternate naming convention).

Options

The options described here are common to all tape drivers. The `c #t #d #` notation in the device special file name derives from *ioscan* output and is described on the manpages for *ioscan(1M)* and *intro(7)*. Options unique to **stape**, **tape1**, and **tape2**, are described later in this manpage, in the **DEPENDENCIES** section.

c #	Instance number assigned by the operating system to the interface card.
t #	Target address on a remote bus (for example, SCSI or HP-IB address)
d #	Device unit number at the target address (for example, SCSI LUN).
w	Writes wait for physical completion of the operation before returning status. The default behavior (buffered mode or immediate reporting mode) requires the tape device to buffer the data and return immediately with successful status.
density	Density or format used in writing data to tape. This field is designated by the following values:
<i>BEST</i>	Highest-capacity density or format will be used, including data compression, if the device supports compression.
<i>NOMOD</i>	Maintains the density used for data previously written to the tape. Behavior using this option is dependent on the type of device. This option is only supported on DDS and 8MM drives.
<i>DDS</i>	Selects one of the known DDS formats; can be used to specify <i>DDS1</i> or <i>DDS2</i> , as required.

<i>DLT</i>	Selects one of the known DLT formats; can be used to specify <i>DLT42500_24</i> , <i>DLT42500_56</i> , <i>DLT62500_64</i> , <i>DLT81633_64</i> , or <i>DLT85937_52</i> , as required.
<i>QIC</i>	Selects one of the known QIC formats; can be used to specify <i>QIC11</i> , <i>QIC24</i> , <i>QIC120</i> , <i>QIC150</i> , <i>QIC525</i> , <i>QIC1000</i> , <i>QIC1350</i> , <i>QIC2100</i> , <i>QIC2GB</i> , or <i>QIC5GB</i> , as required.
<i>D8MM</i>	Selects one of the known 8MM formats; can be used to specify <i>D8MM8200</i> or <i>D8MM8500</i> , as required.
<i>D</i>	Selects a reel-to-reel density; can be used to specify <i>D800</i> , <i>D1600</i> , or <i>D6250</i> , as required.
<i>D3480</i>	Specifies that the device special file communicates with a 3480 device. (There is only one density option for 3480.)
<i>D[#]</i>	Specifies density as a numeric value to be placed in the SCSI mode select block descriptor. This option is not available with HP-IB tape devices. The header file <code><sys/mtio.h></code> contains a list of the standard density codes. The numeric value is used only for density codes which <i>cannot</i> be found in this list.
C[#]	Write data in compressed mode, on tape drives that support data compression. If a number is included, use it to specify a compression algorithm specific to the device. Note, compression is also provided when the density field is set to <i>BEST</i> .
n	No rewind on close. Unless this mode is requested, the tape is automatically rewound upon close.
b	Specifies Berkeley-style tape behavior. When the <i>b</i> is absent, the tape drive follows AT&T-style behavior. The details are described in "Tape Behavioral Characteristics" below.
f#	Specify format (or density) value encoded in the minor number. The meaning of the value is dependent on the type of tape device in use. (Used for short file name notation only.)
i#	Specify an internal Property Table index value maintained by the tape driver, containing an array of configuration options. The contents of this table are not directly accessible. Use the <i>lssf(1M)</i> command to determine which configuration options are invoked. (Used for short file name notation only.)

Sample Tape Device Special File Names

For a QIC150 device at SCSI address 3, card instance 2, with default block size, buffered mode, and AT&T-style with rewind on close, the standard device special file name is `/dev/rmt/c2t3d0QIC150`.

For a device at card instance 1, target 2, LUN 3, with exhaustive mode enabled (see *DEPENDENCIES*), fixed block size of 512 bytes, DDS1 density with compression, AT&T-style with no rewind on close, the standard device file special name is `/dev/rmt/c1t2d3es512DDS1Cn`.

For a system requiring short file names, the same device special file would be named `/dev/rmt/c1t2d3i<#>n`, where `<#>` is an index value selected by the tape driver.

Use the *lssf(1M)* command to determine which configuration options are actually used with any device file. The naming convention defined above should indicate the options used, but device files may be created with any user defined name.

Tape Behavioral Characteristics

When opened for reading or writing, the tape is assumed to be positioned as desired.

When a file opened for writing is closed, two consecutive EOF (End of File) marks are written if, and only if, one or more writes to the file have occurred. The tape is rewound unless the no-rewind mode has been specified, in which case the tape is positioned before the second EOF just written. For QIC devices only one EOF mark is written and the tape is positioned after the EOF mark (if the no-rewind mode has been specified).

When a file open for reading (only) is closed and the no-rewind bit is not set, the tape is rewound. If the no-rewind bit is set, the behavior depends on the *style* mode. For AT&T-style devices, the tape is positioned after the EOF following the data just read (unless already at BOT or Filemark). For Berkeley-style devices, the tape is not repositioned in any way.

Each *read(2)* or *write(2)* call reads or writes the next record on the tape. For writes, the record has the same length as the buffer given (within the limits of the hardware).

During a read, the record size is passed back as the number of bytes read, up to the buffer size specified. Since the minimum read length on a tape device is a complete record (to the next record mark), the number of bytes ignored (for records longer than the buffer size specified) is available in the `mt_resid` field of the `mtget` structure via the `MTIOCGET` call of `ioctl(2)`. Current restrictions require tape device application programs to use 2-byte alignment for buffer locations and I/O sizes. To allow for more stringent future restrictions (4-byte aligned, etc.) and to maximize performance, page alignment is suggested. For example, if the target buffer is contained within a structure, care must be taken that structure elements before the buffer allow the target buffer to begin on an even address. If need be, placing a filler integer before the target buffer will insure its location on a 4-byte boundary.

The ascending hierarchy of tape marks is defined as follows: record mark, filemark (EOF), setmark and EOD (End of Data). Not all devices support all types of tape marks but the positioning within the hierarchy holds true. Each type of mark is typically used to contain one or more of the lesser marks.

When spacing over a number of a particular type of tape mark, hierarchically superior marks (except EOD) do not terminate tape motion and are included in the count. For instance, MTFSR can be used to pass over record marks and filemarks.

Reading an EOF mark is returned as a successful zero-length read; that is, the data count returned is zero and the tape is positioned after the EOF, enabling the next read to return the next record.

DDS devices and the 8mm 8505 device also support setmarks, which are used to delineate a group (set) of files. For the 8mm 8505 setmarks are only supported when the density is set to 8500 plus compression. Reading a setmark is also returned as a zero-length read. Filemarks, setmarks and EOD can be distinguished by unique bits in the `mt_gstat` field.

Spacing operations (back or forward space, setmark, file or record) position past the object being spaced to in the direction of motion. For example, back-spacing a file leaves the tape positioned before the file mark; forward-spacing a file leaves the tape positioned after the file mark. This is consistent with standard tape usage.

For QIC devices, spacing operations can take a very long time. In the worst case, a space command could take as much as 2 hours! While this command is in progress, the device is not accessible for any other commands.

`lseek(2)` type seeks on a magnetic tape device are ignored. Instead, the `ioctl(2)` operations below can be used to position the tape and determine its status.

The header file `<sys/mtio.h>` has useful information for tape handling. The following is included from `<sys/mtio.h>` and describes the possible tape operations:

```
/* mag tape I/O control requests */

#define MTIOCTOP _IOW('m', 1, struct mtop) /* do mag tape op */
#define MTIOCGET _IOR('m', 2, struct mtget) /* get tape status */

/* structure for MTIOCTOP - mag tape op command */
struct mtop {
    short mt_op;          /* operations defined below */
    daddr_t mt_count;     /* how many of them */
};

/* operations */

#define MTWEOF 0 /* write filemark (end-of-file record) */
#define MTFSF 1 /* forward space file */
#define MTBSF 2 /* backward space file */
#define MTFSR 3 /* forward space record */
#define MTBSR 4 /* backward space record */
#define MTREW 5 /* rewind */
#define MTOFFL 6 /* rewind and put the drive offline (may eject) */
#define MTNOP 7 /* no operation, may set status */
#define MTEOD 8 /* DDS, QIC and 8MM only - seek to end-of-data */
#define MTWSS 9 /* DDS and 8MM only - write setmark(s) */
#define MTFSS 10 /* DDS and 8MM only - space forward setmark(s) */
#define MTBSS 11 /* DDS and 8MM only - space backward setmark(s) */

/* structure for MTIOCGET - mag tape get status command */
```

```

struct mtget {
    long    mt_type;      /* type of magtape device */
    long    mt_resid;     /* residual count */

    /* The following two registers are device dependent */

    long    mt_dsreg1;    /* status register (msb) */
    long    mt_dsreg2;    /* status register (lsb) */

    /* The following are device-independent status words */

    long    mt_gstat;     /* generic status */
    long    mt_erreg;     /* error register */
    daddr_t mt_fileno;    /* No longer used - always set to -1 */
    daddr_t mt_blkno;     /* No longer used - always set to -1 */

```

Information for decoding the `mt_type` field can be found in `<sys/mtio.h>`.

Other Tape Status Characteristics

Efficient use of streaming tape drives with large internal buffers and immediate-reporting require the following end-of-tape procedures:

All writes near LEOT (Logical End of Tape) complete without error if actually written to the tape. Once the tape driver determines that LEOT has been passed, subsequent writes do not occur and an error message is returned.

To write beyond this point (keep in mind that streaming drives have already written well past LEOT), simply ask for status using the `MTIOCGGET` ioctl. If status reflects the EOT condition, the driver drops all write barriers. For reel-to-reel devices, caution must be exercised to keep the tape on the reel.

When immediate-reporting is enabled, the `tape1` and `tape2` drivers will drop out of immediate mode and flush the device buffer with every write filemark or write setmark. The stape driver will flush the device buffers when a write filemark or write setmark command is given with the count set to zero.

When immediate-reporting is disabled, the write encountering LEOT returns an error with the tape driver automatically backing up over that record.

When reading near the end-of-tape, the user is not informed of LEOT. Instead, the typical double EOF marks or a pre-arranged data pattern signals the logical end-of-tape.

Since magnetic tape drives vary in EOT sensing due to differences in the physical placement of sensors, any application (such as multiple-tape `cpio(1)` backups) requiring that data be continued from the EOT area of one tape to another tape must be restricted. Therefore, the tape drive type and mode should be identical for the creation and reading of the tapes.

The following macros are defined in `<sys/mtio.h>` for decoding the status field `mt_gstat` returned from `MTIOCGGET`. For each macro, the input parameter `<x>` is the `mt_gstat` field.

<code>GMT_BOT(x)</code>	Returns TRUE at beginning of tape.
<code>GMT_EOD(x)</code>	Returns TRUE if End-of-Data is encountered for DDS, QIC or 8MM.
<code>GMT_EOF(x)</code>	Returns TRUE at an End-of-File mark.
<code>GMT_EOT(x)</code>	Returns TRUE at end of tape.
<code>GMT_IM_REP_EN(x)</code>	Returns TRUE if immediate reporting mode is enabled.
<code>GMT_ONLINE(x)</code>	Returns TRUE if drive is on line.
<code>GMT_SM(x)</code>	Returns TRUE if setmark is encountered.
<code>GMT_WR_PROT(x)</code>	Returns TRUE if tape is write protected.
<code>GMT_COMPRESS(x)</code>	Returns TRUE if data compression is enabled.
<code>GMT_DENSITY(x)</code>	Returns the currently configured 8-bit density value. Supported values are defined in <code><sys/mtio.h></code> .
<code>GMT_QIC_FORMAT(x)</code> and <code>GMT_8mm_FORMAT(x)</code>	Return the same information as does <code>GMT_DENSITY(x)</code> . <code>GMT_DENSITY(x)</code> is preferred because <code>GMT_QIC_FORMAT</code> and <code>GMT_8mm_FORMAT</code> may be obsoleted at some future date.

<i>GMT_D_800(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 800 bpi.
<i>GMT_D_1600(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 1600 bpi.
<i>GMT_D_6250(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 6250 bpi (with or without compression).
<i>GMT_D_6250c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 6250 bpi plus compression.
<i>GMT_D_DDS1(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is DDS1 (with or without compression).
<i>GMT_D_DDS1c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is DDS1 plus compression.
<i>GMT_D_DDS2(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is DDS2 (with or without compression).
<i>GMT_D_DDS2c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is DDS2 plus compression.
<i>GMT_D_DLT_42500_24(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 42500 bpi, 24 track pairs.
<i>GMT_D_DLT_42500_56(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 42500 bpi, 56 track pairs.
<i>GMT_D_DLT_62500_64(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 62500 bpi (with or without compression).
<i>GMT_D_DLT_62500_64c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 62500 bpi plus compression.
<i>GMT_D_DLT_81633_64(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 81633 bpi (with or without compression).
<i>GMT_D_DLT_81633_64c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 81633 bpi plus compression.
<i>GMT_D_DLT_85937_52(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 85937 bpi (with or without compression).
<i>GMT_D_DLT_85937_52c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 85937 bpi plus compression.
<i>GMT_D_3480(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is for a 3480 device (with or without compression).
<i>GMT_D_3480c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is for a 3480 device with compression.
<i>GMT_D_QIC_11(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-11 format.
<i>GMT_D_QIC_24(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-24 format.
<i>GMT_D_QIC_120(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-120 format.
<i>GMT_D_QIC_150(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-150 format.
<i>GMT_D_QIC_525(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-525 format.
<i>GMT_D_QIC_1000(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-1000 format.
<i>GMT_D_QIC_1350(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-1350 format.

<i>GMT_D_QIC_2100(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-2100 format.
<i>GMT_D_QIC_2GB(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-2GB format.
<i>GMT_D_QIC_5GB(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is QIC-5GB format.
<i>GMT_D_8MM_8200(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 8 millimeter 8200 format (with or without compression).
<i>GMT_D_8MM_8200c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 8 millimeter 8200 format with compression.
<i>GMT_D_8MM_8500(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 8 millimeter 8500 format (with or without compression).
<i>GMT_D_8MM_8500c(x)</i>	Returns TRUE if the density encoded in <i>mt_gstat</i> is 8 millimeter 8500 format with compression.
<i>GMT_MEDIUM(x)</i>	Identifies the 8-bit medium type value describing the tape currently loaded into the tape device. The reported value is only valid for QIC and 8mm devices. Supported values are defined in <i><sys/mtio.h></i> .
<i>GMT_QIC_MEDIUM(x)</i>	Returns the same information as does <i>GMT_MEDIUM(x)</i> . <i>GMT_MEDIUM(x)</i> is preferred because <i>GMT_QIC_MEDIUM</i> may be obsoleted at some future date.
<i>GMT_DR_OPEN(x)</i>	Does not apply to any currently supported devices. Always returns FALSE.

HP-UX silently enforces a tape record blocking factor (MAXPHYS) on large I/O requests. For example, a user write request with a length of ten times MAXPHYS will actually reach the media as ten separate records. A subsequent read (with ten times MAXPHYS as a length) will look like a single operation to the user, even though HP-UX has broken it up into ten separate read requests to the driver. The blocking function is transparent to the user during writes. It is also transparent during reads unless:

- The user picks an arbitrary read length greater than MAXPHYS.
- The user attempts to read a third-party tape containing records larger than MAXPHYS.

Since the value for MAXPHYS is relatively large (usually $\geq 256K$ bytes), this is typically not a problem.

The MTNOP operation does not set the device-independent status word.

3480 stacker devices are supported only in auto (that is, sequential-access) mode. To advance to the next tape in the stack, an MTIOCTOP control request specifying an MTOFFL operation should be issued. An MTIOCGET control request should then be issued to determine whether or not the stacker has been successfully advanced. Failure on the MTIOCGET operation (or an offline status) indicates that no more tapes are available in the stacker, the stacker has been ejected, and user intervention is required to load a new stack.

EXAMPLES

Assuming that *fd* is a valid file descriptor, the following example writes two consecutive filemarks on the tape:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtop mtop;

mtop.mt_op = MTWEOF;
mtop.mt_count = 2;
ioctl(fd, MTIOCTOP, &mtop);
```

If *fd* is a valid file descriptor for an open DDS drive, the following example spaces forward to just past the next setmark:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtop mtop;

mtop.mt_op = MTFSS;
mtop.mt_count = 1;
ioctl(fd, MTIOCTOP, &mtop);
```

Given that *fd* is a valid file descriptor for an opened tape device, and that it has just returned 0 from a *read(2)* request. The following system call verifies that the tape has just read a filemark:

```
#include <sys/types.h>
#include <sys/mtio.h>

struct mtget mtget;

ioctl(fd, MTIOCGET, &mtget);
if (GMT_EOF (mtget.mt_gstat)) {
/* code for filemark detection */
}
```

WARNINGS

Density specifications **BEST** (standard naming convention) or **£0** (alternate naming convention) activate data compression on tape devices which support compression. This is also true for the files using the pre-HP-UX 10.0 naming convention which are linked to these files (see "Naming Conventions" above).

This means that a tape written using one of the eight device files (which are automatically created when the system is initialized) on a tape device which supports data compression, will contain compressed data. This tape cannot be successfully read on a tape device which does not support compressed data. For example, a tape written (using one of the eight automatically created device files) on a newer DDS device which supports data compression cannot be read on an older DDS device which does not support data compression.

To accomplish data interchange between devices in a case such as this, a new device file must be manually created using the *mksf(1M)* command. In the above example, the options specified to *mksf(1M)* should include a density option with an argument of **DDS1**, and must not include a compression option.

Use the *mksf(1M)* command instead of the *mknod(1M)* command to create tape device files. As of the 10.0 release, there are more configuration options than will fit in the device file's minor number. Prior to the 10.0 release, it was possible to select configuration options by directly setting the bits in the device special file's minor number using *mknod(1M)*.

As of the 10.0 release, a base set of configuration options are contained in the minor number. Extended configuration options are stored in a table of configuration properties. The minor number may contain an index into the property table, which is maintained by the tape driver and is not directly visible to the user. The *mksf(1M)* command sets the minor number and modifies the property table as needed based on mnemonic parameters passed into the command.

If your device configuration requirements are limited to the base set of options, you need not be concerned with the property table. The base configuration options are as follows:

- hardware address (card instance, target, and unit number)
- density (from the set of pre-defined options listed in *mksf(1M)*)
- compression (using the default compression algorithm)
- rewind or no rewind
- Berkeley or AT&T mode

All other configuration options are extended options that result in use of the property table.

It is recommended that all tape device files be put in the */dev/rmt* directory. All tape device files using extended configuration options *must* be put in the */dev/rmt* directory. This is required for proper maintenance of the property table. Device files using a extended configuration options located outside the */dev/rmt* directory may not provide consistent behavior across system reboots.

Use the *rmsf(1M)* command to clean up unused device files. Otherwise, the property table may overflow and cause the *mksf(1M)* command to fail.

Density codes listed in *<sys/mtio.h>* have device-dependent behaviors. See the hardware manual for your tape device to find which densities are valid. For some devices, these values may be referred to as formats instead of densities.

Use of unbuffered mode can reduce performance and increase media wear.

Reads and writes from/to older (fixed block) devices such as QIC150 must occur at exact multiples of the supported block size.

Write operations on a QIC device can be initiated only at BOT or EOD. QIC devices will not allow writes with the tape positioned in the middle of recorded data.

The offline operation puts the QIC drive offline. The cartridge is not ejected as is done for DDS. To put the drive back online, the cartridge has to be manually ejected and then reinserted.

Sequential-access devices that use the SCSI I/O interface may not always report true media position.

On a 3480 device with data compression enabled, writing of a single record that cannot be compressed to less than 102,400 bytes is not supported.

Note that using the 8200 format on 8500-style 8mm devices will significantly reduce tape capacity, and that only the 8500c-density setting provides support for setmarks.

The maximum I/O request for 8mm devices is limited to 240KB.

DEPENDENCIES

Driver-Specific Options for stape (major number 205)

The following options may be used in creating device special files for tape drives that access the **stape** driver:

- e** Exhaustive mode is enabled (default is disabled).
When exhaustive mode is enabled, the driver will, if necessary, attempt several different configuration options when opening a device. The first attempt follows the minor number configuration exactly, but if that fails, the driver attempts other likely configuration values.
With Exhaustive mode disabled, the driver makes only one attempt to configure a device using the configuration indicated in the minor number.
- p** Specifies a partitioned tape whose currently active partition is partition 1 (closest to BOT (beginning of tape)). Optional partition 1 is closest to BOT for possible use as a volume directory. The default partition without this option is partition 0. If partitioning is unsupported, the entire tape is referred to as partition 0.
- s[#]** Specifies fixed-block mode; the optional number indicates the block size. If the number is not present, the driver selects a default block size appropriate to the device type.

Driver Specific Options for tape1 and tape2 (major number 212)

The following options may be used in creating device special files for tape drives that access the **tape1** and **tape2** drivers:

- o** Diagnostic messages to the console are suppressed.
- z** The tape driver will attempt to mimic the behavior of *RTE* systems; that is, the driver will not do any tape alteration or movement when the device is closed.

Diagnostic Access Naming Convention (tape1 only)

The following alternative naming convention is used to provide diagnostic access for HP-IB devices only:

`/dev/diag/rmt/c#t#d#`

AUTHOR

mt was developed by HP and the University of California, Berkeley.

FILES

<code>/dev/rmt/*</code>	tape device special files
<code>/dev/diag/rmt/*</code>	diagnostic access special files (HP-IB only)
<code><sys/mtio.h></code>	constants and macros for use with tapes
<code>/etc/mtconfig</code>	configuration property table for tapes
<code>/dev/rmt/*config</code>	device files for accessing configuration properties table - for internal use only

SEE ALSO

`dd(1)`, `mt(1)`, `ioctl(2)`, `insf(1M)`, `lssf(1M)`, `mksf(1M)`, `rmsf(1M)`,

Configuring HP-UX for Peripherals

NAME

nfs, NFS - network file system

DESCRIPTION

The Network File System (NFS) allows a client node to perform transparent file access over the network. By using NFS, a client node operates on files residing on a variety of servers and server architectures, and across a variety of operating systems. File access calls on the client (such as read requests) are converted to NFS protocol requests and sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

NFS operates in a stateless manner using remote procedure calls (RPC) built on top of an external data representation (XDR) protocol. The RPC protocol enables version and authentication parameters to be exchanged for security over the network.

A server grants access to a specific file system to clients by adding an entry for that file system to the server's `/etc/exports` file.

A client gains access to that file system using the `mount` command to request a file handle for the file system (see `mount(1M)`). (A file handle is the means by which NFS identifies remote files.) Once a client mounts the file system, the server issues a file handle to the client for each file (or directory) the client accesses. If the file is removed on the server side, the file handle becomes stale (dissociated with a known file), and the server returns an error with `errno` set to [ESTALE].

A server can also be a client with respect to file systems it has mounted over the network; however, its clients cannot directly access those file systems. If a client attempts to mount a file system for which the server is an NFS client, the server returns with `errno` set to [EREMOTE]. The client must mount the file system directly from the server on which the file system resides.

The user ID and group ID mappings must be the same between client and server. However, the server maps UID 0 (the superuser) to UID -2 before performing access checks for a client. This process prevents gaining superuser privileges on remote file systems.

RETURN VALUE

Generally, physical disk I/O errors detected at the server are returned to the client for action. If the server is down or inaccessible, the client receives the message:

NFS: file server not responding: still trying.

The client continues resending the request until it receives an acknowledgement from the server. Therefore, the server can crash or power down, and come back up without any special action required by the client. The client process requesting the I/O will block, but remains sensitive to signals (unless mounted with the `nointr` option) until the server recovers. However, if mounted with the `soft` option, the client process returns an error instead of waiting indefinitely.

AUTHOR

`nfs` was developed by Sun Microsystems, Inc.

SEE ALSO

`exportfs(1M)`, `mount(1M)`, `mount_nfs(1M)`, `nfds(1M)`, `mount(2)`, `checklist(4)`, `exports(4)`.

NAME

null - null file

DESCRIPTION

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

EXAMPLES

To create a zero-length file, use either of the following:

```
cat /dev/null > file
cp /dev/null file
```

FILES

/dev/null

STANDARDS CONFORMANCE

null: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1, POSIX.2

NAME

pckt - Packet Mode module for STREAMS pty (pseudo-terminal)

SYNOPSIS

```
#include <sys/stropts.h>

int ioctl(fd_slave, I_PUSH, "pckt");
```

DESCRIPTION

The **Packet Mode** feature for STREAMS pty devices allows the user process on the master side of the pty device to be informed of state changes in the pty. To enable **Packet Mode** in the STREAMS pty device, the user process must push the **pckt** module onto the master side of the pty with a call to the STREAMS **I_PUSH** *ioctl*(2) system call. When the **pckt** module is pushed onto a STREAMS pty master, certain STREAMS messages going upstream on the master side will get packetized so they can be subsequently retrieved by the master side with a **getmsg** function.

When the user process writes data, the **pckt** module passes the message unchanged downstream on to the next module or driver. When the user process reads data or when the **pckt** module receives certain STREAMS message types, it constructs a packet out of the message for forwarding upstream. To construct a message packet, the module creates an **M_PROTO** message. This **M_PROTO** message contains the original message type in the first data block and the original message in as many data blocks as needed. The user process can then retrieve the **M_PROTO** message with a call to the **getmsg**() function.

The **pckt** module packetizes the following STREAMS message types:

M_DATA, **M_IOCTL**, **M_PROTO**, **M_PCPROTO**, **M_FLUSH**, **M_START**, **M_STOP**, **M_STARTI**, **M_STOPI**, **M_READ**.

All other messages are passed unchanged upstream.

If the message is an **M_FLUSH** message, the **pckt** module looks at the flag and takes the following actions:

- If the flag is **FLUSHW**, the module changes it to **FLUSHR** before creating the **M_PROTO** message and passing the message upstream. This prevents the stream head's read queue from being flushed by the original **M_FLUSH** message.
- If the flag is **FLUSHR**, the module changes it to **FLUSHW** before creating the **M_PROTO** message and passing it upstream. To flush the write queues properly, the module also sends an **M_FLUSH** message with the **FLUSHW** flag set.
- If the flag is **FLUSHRW**, the module changes it to **FLUSHW** before creating the **M_PROTO** message and passing it upstream. To flush the write queues properly, the module also sends an **M_FLUSH** message with the **FLUSHW** flag set.

AUTHOR

pckt(7) was developed by HP and OSF.

SEE ALSO

getmsg(2), **ioctl**(2), **ptm**(7), **pts**(7), **ldterm**(7), **ptem**(7), **streamio**(7).

NAME

ps2, ps2kbd, ps2mouse - PS/2 keyboard/mouse device driver and files

SYNOPSIS

```
#include <sys/ps2io.h>
```

DESCRIPTION

The **ps2** driver allows the use of IBM Personal System/2 (PS/2) compatible keyboards and mouse devices on Hewlett-Packard workstations equipped with PS/2 interface hardware.

On systems with a single interface, PS/2 device file names use the following format:

```
/dev/ps2_n
```

where *n* represents the interface port number, ranging from 0 to 15. For example, the device file **/dev/ps2_1** is used to access port one.

On systems with more than one interface, PS/2 device file names use the following format:

```
/dev/ps2_m.n
```

where *m* represents the interface number, and *n* represents the port number. For example, the device file **/dev/ps2_1.2** is used to access port two on interface one.

At boot time, the **ps2** driver scans all interface ports from port zero to the maximum number of ports implemented and attempts to identify attached PS/2 devices. The **/dev/ps2mouse** device file accesses the first mouse detected by **ps2**. The **/dev/ps2kbd** device file accesses the first keyboard detected by **ps2**.

PS/2 devices are classified as "slow" devices. This means that system calls to **ps2** can be interrupted by caught signals (see *signal(5)*).

The mouse may be placed in one of two output modes. In stream mode, the mouse generates a three-byte report packet in response to mouse movement and/or button presses. These reports can be obtained with the **read()** system call (see *read(2)*). In prompt mode, an **ioctl()** request polls the mouse, returning a three-byte report packet in a buffer whose address is passed as an argument to the **ioctl()** call.

PS/2 keyboards return keycodes that represent key-press and key-release events. Use the Internal Terminal Emulator (ITE) to read ASCII characters from PS/2 keyboards. The ASCII terminal interface used by the ITE is described in *termio(7)*.

The **ps2** driver provides a low-level programming interface to PS/2 keyboards and mice. To access these devices in a hardware independent way, use the X Window programming environment.

System Calls

The **open()** system call gives exclusive access to the specified PS/2 device (see *open(2)*). If a port is open, all **open()** calls made on that port will fail with **errno** set to [EBUSY] (see *errno(2)*).

If an open is attempted on a nonexistent port, the **open()** call fails with **errno** set to [ENXIO].

If no keyboard is detected at system boot and an **open()** is attempted on **/dev/ps2kbd**, or if no mouse is detected at system boot and an **open()** is attempted on **/dev/ps2mouse**, the **open()** call fails with **errno** set to [ENXIO].

Attempts to open an existing **ps2** port with no device connected will succeed.

Upon a successful open, any previously queued input from the device is discarded. Keystrokes are routed to the ITE by default. While a keyboard is open, ITE does not receive keystrokes from that keyboard; until the keyboard device is closed, it has exclusive access to keyboard input.

The file status flags **O_NDELAY** and **O_NONBLOCK** can be set to enable nonblocking reads (see *open(2)*).

read() returns bytes from a PS/2 device. HP-UX maintains a 512-byte buffer for each port. When this buffer is full, additional bytes received from the device are discarded.

If enough buffered data is available to satisfy the entire number of bytes requested, the **read()** call completes successfully, having read all of the data requested and returning the number of bytes read.

If there is not enough buffered data available to satisfy the entire request, but at least one byte is available, the **read()** call completes successfully, having read all available data and returning the number of bytes actually read.

(Series 700 Only)

If both file status flags `O_NDELAY` and `O_NONBLOCK` are clear and no data is available, the `read()` call blocks until data becomes available or a signal is received.

If the file status flag `O_NDELAY` is set and no data is available, the `read()` call returns zero instead of blocking.

If the file status flag `O_NONBLOCK` is set and no data is available, the `read()` call returns `-1` with `errno` set to `[EAGAIN]` (see *errno(2)*).

The `write()` system call is not supported by **ps2**.

The `select()` system call can be used to determine if data is currently available to be read from a **ps2** port. Using `select()` for write or for exception conditions always returns a false indication in the file descriptor bit masks (see *select(2)*).

The `ioctl()` system call is used to perform special operations on PS/2 mouse and keyboard devices (see *ioctl(2)*). The set of **ps2** driver `ioctl()` requests are divided into three groups: general requests to both mouse and keyboard, keyboard-specific requests, and mouse-specific requests. Mouse-specific requests used on keyboards, and keyboard-specific requests used on mice, fail, returning `-1` with `errno` set to `[EINVAL]`.

Any `ioctl()` request (except `PS2_PORTSTAT`) used on a port not connected to a PS/2 device will time out, returning `-1` with `errno` set to `[EIO]`.

All `ioctl()` system calls use the following syntax:

```
int ioctl(int fildes, int request, char *arg);
```

All requests that require parameters or return data use a 4-byte unsigned character buffer addressed by the *arg* argument.

The *request* codes that follow are defined in `<sys/ps2io.h>`.

General ioctl() Requests for Both Keyboard and Mouse

PS2_PORTSTAT

Return driver status information.

Two bytes of data are returned in the character buffer addressed by *arg*.

Byte 0, which indicates the type of connected device, can have four possible values:

PS2_NONE	No device is detected.
PS2_MOUSE	Mouse is detected.
PS2_KEYBD	Keyboard is detected.
PS2_UNKNOWN	Unknown device is detected.

Byte 1 contains bit flags for various pieces of driver information. The following bit masks for this byte are defined in the file `/usr/include/sys/ps2io.h`:

INTERFACE_HAS_ITE	If set, the interface containing this port is used by the Internal Terminal Emulator (ITE) for keyboard input.
PORT_HAS_FIRST_KEYBD	If set, this port is connected to the first keyboard detected by the driver.
PORT_HAS_FIRST_MOUSE	If set, this port is connected to the first mouse detected by the driver.

All other bits are currently unused, and are cleared to zero.

PS2_DISABLE

Disable a PS/2 device.

Further output from the device is prevented by the device itself. This request does not use *arg*. Certain devices perform actions in addition to disabling themselves.

The keyboard resets its internal state to the default state, stops scanning the keys, and waits for further commands.

The mouse stops transmission of reports, and then disables itself.

(Series 700 Only)

PS2_ENABLE	Enable a PS/2 device Transmissions from the device are enabled. This request does not use <i>arg</i> .
PS2_IDENT	Identify a PS/2 device. A value identifying the type of device is returned in the 4-byte buffer addressed by <i>arg</i> . The keyboard returns two bytes (<i>arg</i> [0]=0xAB and <i>arg</i> [1]=0x83). The mouse returns one byte (<i>arg</i> [0]=0x00).
PS2_SETDEFAULT	Set the device to its default (power-up) state. The device is returned to its default internal state. This request does not use <i>arg</i> .
PS2_RESET	Reset a PS/2 device. The device is told to execute its internal reset routine and execute its power-up test. The result of the power-up test is returned in the 4-byte buffer addressed by <i>arg</i> . The mouse returns two bytes to indicate a successful reset (<i>arg</i> [0]=0xAA and <i>arg</i> [1]=0x00). The keyboard returns one byte (<i>arg</i> [0]=0xAA).

Keyboard-Specific ioctl() Requests

PS2_SCANCODE	Select the keyboard scancode set The scancode set to be used by the keyboard is passed as the first byte of the buffer addressed by <i>arg</i> . The following are valid values for this byte: <table> <tr> <td>SCANCODE_1</td><td>Selects scancode set 1.</td></tr> <tr> <td>SCANCODE_2</td><td>Selects scancode set 2.</td></tr> <tr> <td>SCANCODE_3</td><td>Selects scancode set 3.</td></tr> <tr> <td>GET_SCANCODE</td><td>Returns the scancode used.</td></tr> </table> When GET_SCANCODE is specified, the scancode used by the keyboard is returned as the first byte of the character buffer addressed by <i>arg</i> . Some keyboards do not support all scancode sets.	SCANCODE_1	Selects scancode set 1.	SCANCODE_2	Selects scancode set 2.	SCANCODE_3	Selects scancode set 3.	GET_SCANCODE	Returns the scancode used.
SCANCODE_1	Selects scancode set 1.								
SCANCODE_2	Selects scancode set 2.								
SCANCODE_3	Selects scancode set 3.								
GET_SCANCODE	Returns the scancode used.								
PS2_ALL_TMAT	Set all keys to typematic behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used. The typematic rate and delay are set via the PS2_RATEDELAY <i>ioctl</i> () request.								
PS2_ALL_MK	Set all keys to make-only behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used.								
PS2_ALL_MKBRK	Set all keys to make/break behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used.								
PS2_ALL_TMAT_MKBRK	Set all keys to typematic make/break behavior. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The <i>arg</i> parameter is not used. The typematic rate and delay are set via the PS2_RATEDELAY <i>ioctl</i> () request.								
PS2_KEY_TMAT	Set typematic behavior for an individual key. The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by <i>arg</i> . This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. The typematic rate and delay are set via the PS2_RATEDELAY <i>ioctl</i> () request. Because keyboards might be left in a disabled state after this request, the PS2_ENABLE request should be								

(Series 700 Only)

performed after **PS2_KEY_TMAT**.

PS2_KEY_MAKE

Set make-only behavior for an individual key.

The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by *arg*. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. Because keyboards might be left in a disabled state after this request, the **PS2_ENABLE** request should be performed after **PS2_KEY_MAKE**.

PS2_KEY_MKBRK

Set make/break for an individual key.

The key code from scancode set 3 for the individual key is passed as the first byte in the character buffer addressed by *arg*. Make/break behavior will be set for this key. This request can be made when the keyboard is using any scancode set; however, it affects only the operation of scancode set 3. Because keyboards might be left in a disabled state after this request, the **PS2_ENABLE** request should be performed after **PS2_KEY_MKBRK**.

PS2_INDICATORS

Set the state of keyboard indicators, Num Lock, Caps Lock, and Scroll Lock, according to the value passed in the first byte of the character buffer addressed by *arg*.

The indicators are bit-mapped as follows:

NONE_LED	No indicators active
CAPS_LED	Caps Lock indicator active
NUM_LED	Num Lock indicator active
SCROLL_LED	Scroll Lock indicator active

PS2_RATEDELAY

Set the rate and delay for all typematic keys by specifying the value passed as the first byte in the character buffer addressed by *arg*.

Bits zero through four give the rate. Bits five and six give the delay. Bit seven (the most significant bit) is unused and should be set to zero. The delay in milliseconds is determined by the following equation, where *X* is the numeric value of bits five through six:

$$\text{delay} = (1 + X) * 250 \quad (+/- 20\%)$$

The period (interval from one output key code to the next) in seconds is determined by the following equation, where *Y* is the numeric value of bits zero through two, and *Z* is the numeric value of bits three through four:

$$\text{period} = (8 + Y) * (2^Z) * 0.00417 \quad (+/- 20\%)$$

The typematic rate (expressed in make codes per second) is one for each period using the above equation. The default typematic rate is 10.9 characters per second. The default delay is 500 milliseconds.

Mouse-Specific ioctl() Requests**PS2_SAMPLERATE**

Set the mouse sampling rate used in stream mode by specifying the value passed as the first byte in the character buffer addressed by *arg*.

Seven specific rates are supported:

SAMPLE_10	10 reports/second maximum
SAMPLE_20	20 reports/second maximum
SAMPLE_40	40 reports/second maximum
SAMPLE_60	60 reports/second maximum
SAMPLE_80	80 reports/second maximum
SAMPLE_100	100 reports/second maximum
SAMPLE_200	200 reports/second maximum

The default rate is 100 reports/second maximum. This request updates the mouse sampling rate only in stream mode. If the mouse is in prompt mode, this request is ignored.

PS2_PROMPTMODE

Put mouse into prompt mode.

(Series 700 Only)

In prompt mode, the mouse updates its internal values due to movement or button presses, but issues reports only in response to the **PS2_REPORT ioctl()** request. The *arg* parameter is not used.

PS2_REPORT

Obtain a prompt mode mouse report.

This request polls the mouse, obtaining a three-byte report returned in the character buffer addressed by the *arg* parameter. The report has the following format:

Byte 1 A bit map of buttons, signs, and overflows

Bit 0	Left button (1=depressed)
Bit 1	Right button (1=depressed)
Bit 2	Center button (1=depressed)
Bit 3	Always 1
Bit 4	X data sign (1=negative)
Bit 5	Y data sign (1=negative)
Bit 6	X data overflow (1=overflow)
Bit 7	Y data overflow (1=overflow)

Byte 2 X-coordinate data byte

Byte 3 Y-coordinate data byte

The X and Y coordinate values are expressed in two's complement. The scaling behavior specified via the **PS2_2TO1_SCALING ioctl()** request does not apply to reports obtained with the **PS2_REPORT ioctl()** request. **PS2_2TO1_SCALING** affects only reports sent in stream mode.

PS2_STREAMMODE

Put mouse into stream mode.

When in stream mode, the mouse sends a three-byte report whenever the mouse is moved, or a button is pressed or released since the last report. The maximum report rate is set with the **PS2_SAMPLERATE ioctl()** request. If a button is both pressed and then released within a sample interval, it will be reported as pressed at the end of that interval.

The stream-mode reports are obtained via the **read()** system call (see *read(2)*). The format of the report is identical to reports returned by the **PS2_REPORT ioctl()** request described above.

When in stream mode, the **PS2_DISABLE** request must be sent prior to any other **ioctl()** requests.

The *arg* parameter is not used.

PS2_STATUS

Obtain mouse status.

This request polls the mouse, obtaining a three-byte report returned in the character buffer addressed by the *arg* parameter.

The status report has the following format:

Byte 1 A bit map of buttons and mouse internal state

Bit 0	Right button (1=depressed)
Bit 1	Center button (1=depressed)
Bit 2	Left button (1=depressed)
Bit 3	Always 0
Bit 4	If 0, scaling 1:1; if 1, scaling 2:1
Bit 5	If 0, disabled; if 1, enabled
Bit 6	If 0, stream mode; if 1, prompt mode
Bit 7	Always 0

Byte 2 Current resolution setting

Byte 3 Current sampling rate

PS2_RESOLUTION

Set mouse resolution for X and Y coordinate values by specifying the value passed as the first byte in the character buffer addressed by *arg*. Four discrete resolutions are supported:

(Series 700 Only)

Resolution	200 DPI	320 DPI
RES_1	1 count/mm	1 count/mm
RES_2	2 count/mm	3 count/mm
RES_3	4 count/mm	6 count/mm
RES_4	8 count/mm	12 count/mm

PS2_2TO1_SCALING

Set mouse scaling at 2 to 1. The X and Y coordinate values returned in stream-mode reports are doubled, except for absolute values less than six, which are converted to new values in a nonlinear fashion. The conversion is detailed in this table:

Mouse Internal Value	Converted Value
0	0
+ - 1	+ - 1
+ - 2	+ - 1
+ - 3	+ - 3
+ - 4	+ - 6
+ - 5	+ - 9
All other <i>n</i>	2 * <i>n</i>

This conversion does not apply to reports obtained via the **PS2_REPORT_IOCTL()** request.

The *arg* parameter is not used.

PS2_1TO1_SCALING

Set mouse scaling at 1 to 1.

The X and Y values returned in mouse reports are not scaled. This request does not use the *arg* parameter.

ERRORS

If a system call fails, as noted above in the DESCRIPTION section **errno** is set to one of the following values:

[EBUSY]	The specified PS/2 device is already opened.
[EFAULT]	A bad address was detected while attempting to use an argument to a system call.
[EINTR]	A signal interrupted an open() , read() , or ioctl() system call.
[EINVAL]	An invalid parameter was detected by ioctl() .
[EIO]	A hardware or software error occurred while executing an ioctl() system call.
[ENODEV]	write() is not implemented for PS/2 devices.
[ENXIO]	No device is present at the specified address.

EXAMPLES

Assume that *fildev* is a valid file descriptor for a **ps2** port connected to a keyboard. The first example blinks the keyboard indicators, selects scancode set 3, and loops forever while printing keycodes.

```
#include <sys/ps2io.h>

unsigned char kbdbuf[4]; /* buffer for ioctl operations */
unsigned char inchar;    /* keycode read */

/* flash the LED indicators */
kbdbuf[0] = CAPS_LED | SCROLL_LED | NUM_LED; /* all on */
if( ioctl( fildev, PS2_INDICATORS, &kbdbuf) < 0){
    perror("ioctl PS2_INDICATORS failed");
    exit(1);
}
printf("Indicators on\n");
sleep(1);

kbdbuf[0] = NONE_LED; /* all off */
if( ioctl( fildev, PS2_INDICATORS, &kbdbuf) < 0){
    perror("ioctl PS2_INDICATORS failed");
}
```



```

    exit(1);
}
printf("Indicators off\n");

/* use scancode set 3 */
kbdbuf[0] = SCANCODE_3;
if( ioctl( fildes, PS2_SCANCODE, &kbdbuf) < 0){
    perror("ioctl PS2_SCANCODE failed");
    exit(1);
}

/* identify our scancode set */
kbdbuf[0] = GET_SCANCODE;
if( ioctl( fildes, PS2_SCANCODE, &kbdbuf) < 0){
    perror("ioctl PS2_SCANCODE failed");
    exit(1);
}
printf("Keyboard reports it is using scancode set %d\n",
       (unsigned int) kbdbuf[0]);

/* now, loop forever while printing keycodes */
while( 1){
    read( fildes, &inchar, 1);
    printf("Keycode: %x\n", (unsigned int)inchar);
}

```

The following example puts the mouse in stream mode, sets the report limit to 80 per second, enables the mouse, and then loops forever printing mouse reports. Assume that *fildes* is a valid file descriptor for a ps2 port connected to a mouse.

```

#include <sys/ps2io.h>

unsigned char buf[3];          /* mouse report buffer */
unsigned char ioctl_buf[4];   /* mouse ioctl buffer */

/* first, disable the mouse */
if (ioctl( fildes, PS2_DISABLE) < 0){
    perror("ioctl PS2_DISABLE failed\n");
    exit(1);
}
printf("Mouse disabled\n");

/* Put mouse in stream mode */
if (ioctl( fildes, PS2_STREAMMODE) < 0){
    perror("ioctl PS2_STREAMMODE failed\n");
    exit(1);
}
printf("Mouse in stream mode\n");

/* set samplerate */
ioctl_buf[0] = SAMPLE_80;
if (ioctl( fildes, PS2_SAMPLERATE, ioctl_buf) < 0){
    perror("ioctl PS2_SAMPLERATE failed\n");
    exit(1);
}
printf("Mouse sample rate set to SAMPLE_80\n");

/* Enable mouse */
if (ioctl( fildes, PS2_ENABLE) < 0){
    perror("ioctl PS2_ENABLE failed\n");
    exit(1);
}
printf("Mouse enabled.\n");

```

(Series 700 Only)

```

for (;;) {
    if (read(fildes, &buf[0], 1) != 1){
        perror("Read of report byte 1 failed");
        return 1;
    }
    if (read(fildes, &buf[1], 1) != 1){
        perror("Read of report byte 2 failed");
        return 1;
    }
    if (read(fildes, &buf[3], 1) != 1){
        perror("Read of report byte 3 failed");
        return 1;
    }
    printf("mouse: 0x%02x, %d %d\n", buf[0], buf[1], buf[2]);
}

```

AUTHOR

ps2 was developed by the Hewlett-Packard Company.

PS/2 and Personal System/2 are registered trademarks of International Business Machines, Incorporated, in the U.S. and other countries.

FILES

```

/usr/include/sys/ps2io.h
/dev/ps2_[0-15]
/dev/ps2_*. [0-15]
/dev/ps2mouse
/dev/ps2kbd

```

SEE ALSO

close(2), errno(2), fcntl(2), ioctl(2), open(2), read(2), select(2), signal(5), termio(7).

SoftPC User's Guide

SoftPC Installation Guide

Sun System Administrators Guide for the HP700/RX

NAME

ptem - STREAMS pty (pesudo-terminal) Emulation module

SYNOPSIS

```
#include <sys/stropts.h>

int ioctl(fd_slave, I_PUSH, "ptem");
```

DESCRIPTION

ptem is a STREAMS module that emulates a terminal when used in conjunction with **ldterm** (STREAMS line discipline) and **pts** (STREAMS slave pty driver). The **ptem** module normally sits above **pts** and below **ldterm**. The user process must push the **ptem** module onto the slave side of the pty with a call to the STREAMS **I_PUSH ioctl(2)** system call before **ldterm** is pushed. **ptem** is responsible for processing all of the terminal **ioctl** commands that are passed downstream from **ldterm** or from **ptm** (STREAMS pty master driver).

ldterm and **ptem** together provide a real terminal behavior for the STREAMS pty slave. However, some of the terminal **ioctl** commands are ignored and cause only an acknowledgement of the command since there is no real terminal or modem in the pty subsystem. In fact, none of the flags in the **c_cflag** field of the **termio** or **termios** structures, (which is used by the **TCSETA** or **TCSETS ioctls**, respectively), have any effect on the pty except if the baud rate is set to zero. Setting the baud rate to zero will have the effect of hanging up the pty connection. Similarly, the parity or delay flags in the **c_iflag** field will not have any effect at all on the pty.

As a summary, the **ptem** module performs the following tasks:

- The following **ioctls** are processed, if appropriate, and acknowledged by sending an **M_IOCACK** message upstream when they are received on **ptem**'s write queue:
TCSETA, **TCSETAW**, **TCSETAF**, **TCSETS**, **TCSETSW**, **TCSETSF**, **TCGETA**, **TCGETS**, and **TCSBRK**.
- Keeps track of the window size needed for the **TIOCSWINSZ**, **TIOCGWINSZ**, and **JWINSIZE ioctls**.
- Upon receiving any other **ioctl** on its write queue, **ptem** acknowledges them negatively by sending an **M_IOCNAK** message upstream.
- The following **ioctls** are passed downstream by **ptem** after they have been processed:
TCSETA, **TCSETAW**, **TCSETAF**, **TCSETS**, **TCSETSW**, **TCSETSF**, **TCSBRK**, and **TIOCSWINSZ**.
- Any **M_IOCNAK** message that is received on **ptem**'s read queue will be freed in case the **pckt** module is not pushed on the **ptm** and the above **ioctls** get to the pty master STREAMS head, which would then send an **M_IOCNAK** message downstream.
- When **ptem** is opened and all conditions for setting up a controlling terminal are met, it sends an **M_SETOPTS** message (with the **SO_ISATTY** flag set) upstream to the STREAMS head to allocate a controlling terminal.
- Upon receiving an **M_IOCTL** message of type **TCSBRK** on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_BREAK** message upstream.
- When an **ioctl** message is received on its write queue to set the baud rate to zero (e.g. **TCSETA** with **CBAUD** set to **B0**), **ptem** sends an **M_IOCACK** message upstream and a zero-length message downstream to be read by the pty master process.
- When an **M_IOCTL** message of type **TIOCSIGNAL** is received on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_PCISIG** message upstream with the signal number set to the same value used in the **M_IOCTL** message.
- When an **M_IOCTL** message of type **TIOCREMOTE** is received on its read queue, **ptem** sends an **M_IOCACK** message downstream and an **M_CTL** message (with **ioc_cmd** set to **MC_DO_CANON** or **MC_NO_CANON**) upstream to enable or disable the input processing on **ldterm**.
- When an **M_DELAY** message is received on its read or write queue, **ptem** simply discards the message without any action.
- When an **M_IOCTL** message of type **JWINSIZE** is received on its write queue and if the values in the **jwinsize** structure in **ptem** are not zero, **ptem** sends an **M_IOCACK** message

upstream with the **jwinsize** structure. If the values are zero, **p_{tem}** sends an **M_IOCNAK** message upstream.

- When an **M_IOCTL** message of type **TIOCGWINSZ** is received on its write queue and if the values in the **winsize** structure in **p_{tem}** are not zero, **p_{tem}** sends an **M_IOCACK** message upstream with the **winsize** structure. If the values are zero, **p_{tem}** sends an **M_IOCNAK** message upstream.
- When an **M_IOCTL** message of type **TIOCSWINSZ** is received in its write queue, **p_{tem}** saves the information passed to it in the **winsize** structure and sends an **M_PCSIG** (with the signal number set to **SIGWINCH**) upstream to the pty slave process if the window size is changed.
- When an **M_IOCTL** message of type **TIOCGWINSZ** is received on its read queue and if the values in the **winsize** structure in **p_{tem}** are not zero, **p_{tem}** sends an **M_IOCACK** message downstream with the **winsize** structure. If the values are zero, **p_{tem}** sends an **M_IOCNAK** message downstream.
- When an **M_IOCTL** message of type **TIOCSWINSZ** is received in its read queue, **p_{tem}** saves the information passed to it in the **winsize** structure and sends an **M_PCSIG** (with the signal number is set to **SIGWINCH**) upstream to the pty slave process if the window size is changed.
- All other messages not mentioned above are passed to the next module or driver.

AUTHOR

p_{tem} was developed by HP.

SEE ALSO

ioctl(2), **streamio**(7), **ptm**(7), **pts**(7), **ldterm**(7).

NAME

ptm - STREAMS master pty (pseudo-terminal) driver

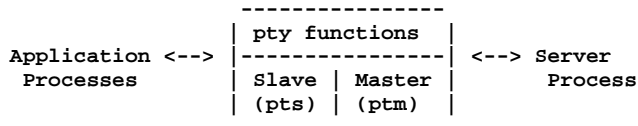
SYNOPSIS

```
#include <sys/stropts.h>
#include <sys/ptyio.h>
#include <sys/strtio.h>

int open("/dev/ptmx", O_RDWR);
```

DESCRIPTION

A pseudo-terminal (pty) consists of a tightly-coupled pair of character devices, called the master device and slave device. The pty master and slave device drivers work together to simulate a terminal connection where the master provides a connection to the pseudo terminal server process and the slave provides a terminal device special file access for the terminal application processes, as depicted below:



The slave driver, **pts** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has another process manipulating it through the master side of the pty. Data written on the master device is given to the slave device as input and data written on the slave device is presented as input on the master device.

In order to use the STREAMS pty subsystem, a node for the master pty driver **/dev/ptmx** and *N* number of slave pty devices must be installed (see *pts(7)* for details on slave pty). There are no nodes in the file system for each individual master device. Rather, the master driver is set up as a STREAMS clone driver (see *clone(7)*) with its major device number set to the major for the clone driver and its minor device number set to the major for the **ptm** driver. The master driver is opened using the **open()** system call with **/dev/ptmx** as the device file parameter. The clone open finds the next available minor number for the master device. The master device is available only if it and its corresponding slave device are not already opened. Only one open is allowed on a master device whereas multiple open are allowed on the slave device. When the master device is opened, the corresponding slave device is automatically locked out (see *pts(7)* on how to unlock the slave and obtain the slave device name). After both the master and slave have been opened, the user has two file descriptors which represent the end points of a full duplex connection composed of two streams. These two streams are automatically connected by the master and slave devices when they are opened. The user may then push the necessary modules on the master and slave streams (e.g., **ptem** and **ldterm**, on **pts** for terminal semantics, and **pckt** on **ptm** for Packet Mode feature).

The master and slave drivers pass all STREAMS messages to their adjacent drivers. Only the **M_FLUSH** message needs some special processing because the read queue of the master is connected to the write queue of the slave and vice versa. Hence, the **FLUSHR** flag is changed to **FLUSHW** flag and vice versa whenever a **M_FLUSH** message travels across the master-slave link. When the master device is closed, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the **errno** [ENXIO] when attempting a **write()** system call on the slave device but it will be able to read any data remaining on the slave stream. Finally, when all the data have been read, the **read()** system call will return 0 (zero) indicating that the slave can no longer be used. On the last close of the slave device, a zero-length **M_DATA** message is sent to the corresponding master device. When the application on the master side issues a **read()** or **getmsg()** system calls and a 0 is returned. The user of the master device decides whether to close the master device file which will dismantle the streams on the master side. If the master device remains opened, the corresponding slave device can be opened and used again by another user.

Unlike the slave device, the master device does not act like a terminal. If **O_NDELAY** or **O_NONBLOCK** is set, a read on the master device returns -1 with **errno** set to [EAGAIN] if no data is available, and a write returns -1 with **errno** set to [EAGAIN] if there is internal flow control on the stream.

The master **ptm** driver supports the following **ioctl()** requests:

ISPTM Determines whether the file descriptor is that of an open master device. On success, it returns the major and minor number (type `dev_t`) of the master device which can be used to determine the name of the corresponding slave device. On failure, it returns `-1` with `errno` set to `[EINVAL]`. **ISPTM** on HP-UX can return valid device number with negative value. For example, with major number of the STREAMS pty master being `0x9c`, **ICPTM** will return `0x9C000000` which is a negative number. Therefore, it is imperative that applications check for an explicit `-1` instead of `"< 0"` (less than 0) on the return value.

ISPTM is used by functions `grantpt()`, `unlockpt()`, and `ptsname()`. User applications normally do not need to invoke this ioctl. The format of this ioctl is:

```
int ioctl(master_fd, ISPTM, 0)
```

UNLKPT Unlocks the master and the corresponding slave devices. On success, it returns 0. On failure, it returns `-1` with `errno` set to `[EINVAL]`. **UNLKPT** is used by function `unlockpt()`. User applications normally do not need to invoke this ioctl. The format of this ioctl is:

```
int ioctl(master_fd, UNLKPT, 0)
```

TIOCREMOTE This ioctl puts the STREAMS pty in and out of Remote Mode. When Remote Mode is on, input data will be flow-controlled and passed through `ldterm` without any input processing regardless of the terminal mode. When the pty master driver receives this ioctl, it will send an `M_CTL` message downstream to `ldterm` via `ptm`, `pts`, and `ptem`. The command in the `M_CTL` message is set to `MC_NO_CANON` or `MC_DO_CANON` depending whether to turn on or off the Remote Mode. The format of this ioctl is:

```
int ioctl(master_fd, TIOCREMOTE, argument)
```

where the argument is set to 1 to turn on Remote Mode and 0 to turn it off. Remote Mode is normally used when doing remote line editing in a window manager, or whenever flow-controlled input is required. Each write to the master device produces a record boundary for the process reading the slave devices. In normal usage, a write of data is like the data typed as a line on the terminal; a write of 0 (zero) bytes is like typing an **EOF** (End-of-File) character.

TIOCSIGNAL This ioctl allows the master process to send a signal to the slave process. The format of this ioctl is:

```
int ioctl(master_fd, TIOCSIGNAL, argument)
```

where the argument is the signal number as defined in the header file `<sys/signal.h>`. For example the master process can send an **SIGINT** signal to the slave process by doing:

```
ioctl(master_fd, TIOCSIGNAL, SIGINT)
```

AUTHOR

ptm was developed by HP and OSF.

FILES

`/dev/ptmx` Streams pty master clone device
`/dev/pts/N` Streams pty slave devices ($0 \leq N < \text{NSTRPTY}$), where **NSTRPTY** is a kernel tunable parameter which can be changed via SAM.

SEE ALSO

`insf(1M)`, `getmsg(2)`, `ioctl(2)`, `open(2)`, `read(2)`, `write(2)`, `grantpt(3C)`, `ptsname(3C)`, `unlockpt(3C)`, `clone(7)`, `ldterm(7)`, `pckt(7)`, `ptem(7)`, `pts(7)`, `streamio(7)`, `termio(7)`.

NAME

pts - STREAMS slave pty (pseudo-terminal) driver

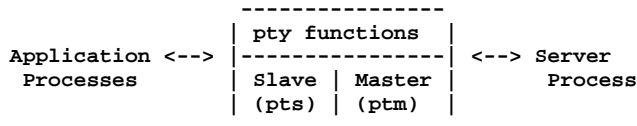
SYNOPSIS

```
#include <sys/stropts.h>
#include <sys/termios.h>
#include <sys/strtio.h>

int open("/dev/pts/N", O_RDWR);
```

DESCRIPTION

A pseudo-terminal (pty) consists of a tightly-coupled pair of character devices, called the master device and slave device. The pty master and slave device drivers work together to simulate a terminal connection where the master provides a connection to the pseudo terminal server process and the slave provides a terminal device special file access for the terminal application processes, as depicted below:



The slave driver, **pts** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has another process manipulating it through the master side of the pty. Data written on the master device is given to the slave device as input and data written on the slave device is presented as input on the master device.

In order to use the STREAMS pty subsystem, a node for the master pty driver **/dev/ptmx** and *N* number of slave pty devices must be installed (see *ptm(7)* for more details on master pty). When the master device is opened, the corresponding slave device is automatically locked out. No user can open that slave device until its permissions are changed (via the **grantpt()** function) and the device is unlocked (via the **unlockpt()** function). The user then call the **ptsname()** function to obtain the name of the slave device and invoke the **open()** system call to open the slave device. Although only one open is allowed on a master device, multiple opens are allowed on the slave device. After both the master and slave have been opened, the user has two file descriptors which represent the end points of a full duplex connection composed of two streams that are automatically connected by the master and slave devices when they are opened. The user may then push the desired modules (for example, **ptem** and **ldterm**, on **pts** for terminal semantics and **pckt** on **ptm** for Packet Mode feature).

The master and slave drivers pass all STREAMS messages to their adjacent drivers. Only the **M_FLUSH** message needs some special processing because the read queue of the master is connected to the write queue of the slave and vice versa. For example, the **FLUSHR** flag is changed to **FLUSHW** flag and vice versa whenever a **M_FLUSH** message travels across the master-slave link. When the master device is closed, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the **errno** [ENXIO] when attempting a **write()** system call to the slave device file but it will be able to read any data remaining in the slave stream. Finally, when all the data has been read, the **read()** system call will return 0, indicating that the slave can no longer be used. On the last close of the slave device, a zero-length **M_DATA** message is sent to the corresponding master device. When the application on the master side issues a *read(2)* or *getmsg(2)* system calls, a 0 (zero) is returned. The user of the master device may decide to close the master device file, which dismantles the stream on the master side. If the master device remains opened, the corresponding slave device can be opened and used again by another user.

EXAMPLES

The following example shows how a STREAMS pty master and slave devices are typically opened.

```
int fd_master, fd_slave;
char *slave;
...

fd_master = open("/dev/ptmx", O_RDWR);
grantpt(fd_master);
unlockpt(fd_master);
```

```

slave = ptsname(fd_master);
fd_slave = open(slave, O_RDWR);
ioctl(fd_slave, I_PUSH, "ptem");
ioctl(fd_slave, I_PUSH, "ldterm");

```

AUTHOR

pts was developed by HP and OSF.

FILES

/dev/**ptmx** Streams pty master clone device
 /dev/**pts**/*N* Streams pty slave devices ($0 \leq N < \text{NSTRPTY}$), where **NSTRPTY** is a kernel tunable parameter which can be changed via SAM (see *sam*(1M)).

SEE ALSO

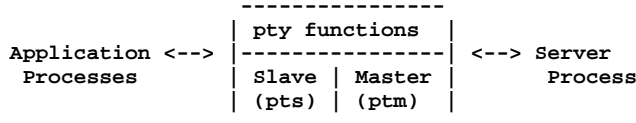
insf(1M), sam(1M), getmsg(2), ioctl(2), open(2), read(2), write(2), grantpt(3C), ptsname(3C), unlockpt(3C), ldterm(7), ptem(7), ptm(7), streamio(7), termio(7).

NAME

pty - pseudo terminal driver

DESCRIPTION

The **pty** driver provides support for a device-pair termed a pseudo terminal. A pseudo terminal is a pair of character devices, a master device and a slave device. The slave device provides to application processes an interface identical to that described in *termio*(7). Unlike all other devices that provide the interface described in *termio*(7), the slave device does not have a hardware device behind it. Instead, it has another process manipulating it through the master half of the pseudo terminal. Thus anything written on the master device is given to the slave device as input, and anything written on the slave device is presented as input on the master device.

**Open and Close Processing**

The slave side of the **pty** interprets opening or closing the master side as a modem connection or disconnection on a real terminal. Only one open to the master side of a **pty** is permitted. An attempt to open an already open master side returns **-1** and sets the external variable **errno** to [EBUSY]. An attempt to open the master side of a **pty** that has a slave with an open file descriptor returns **-1** and sets **errno** to [EBUSY]. The potential problem of **ptys** being found busy at opens can be avoided by using the *clone open* functionality discussed in the next section.

An attempt to open a nonexistent **pty** returns **-1** and sets **errno** to [ENXIO]. If **O_NDELAY** is not specified, opens on the slave side hang until the master side is opened. If **O_NDELAY** is specified, opens on the slave side return error if the master side is closed. Any **ioctl()** or **write()** request made on the slave side of a **pty** after the master side is closed returns **-1** and sets the external variable **errno** to [EIO]. A **read()** request made on the slave side of a **pty** after the master side is closed returns 0 bytes. Closing the master side of a **pty** sends a **SIGHUP** hangup signal to the tty process group number of the corresponding slave side and flushes pending input and output.

Clone Open

In typical **pty** usage, there is no preference among **pty** pairs. Thus, it is useful to be able to issue a single **open()** that internally opens any available **pty**. An open on **/dev/ptym/clone** returns an open file descriptor of a free master **pty** device. If there are no free devices, the open returns **-1** and sets **errno** to [EBUSY]. The name of the slave device corresponding to the opened master device can be found through a **ptsname()** request.

Processing ioctl() Requests

By default, any **ioctl()** request defined by *termio*(7) is recognized by both the master and slave sides of a **pty**. These **ioctl()** requests are processed by the **pty** driver as specified by *termio*(7). In addition, the **ioctl()** requests defined below are recognized by the master side of a **pty**. The slave side only recognizes **ioctl()** requests defined by *termio*(7). An **ioctl()** request made on the slave side of a **pty** after the master side is closed returns **-1** and sets the external variable **errno** to [EIO]. An **ioctl()** request not recognized by the **pty** returns **-1** and sets the external variable **errno** to [EINVAL]. Note that some of the master-side-only **ioctl()** requests affect which **ioctl()** requests are recognized by the master and slave side of the **pty**. These master-side-only **ioctl()** requests also affect the way recognized **ioctl()** requests, **open()** requests, and **close()** requests are processed by the **pty** driver.

The following **ioctl()** requests, defined in **<sys/ptyio.h>**, apply only to the master side of **pty**:

TIOCSIGSEND

Cause a signal to be sent from the slave side of the **pty** to the current tty process group of the slave side. The value of the parameter is taken to be the signal number sent. An [EINVAL] error is returned and no signal is sent if the specified signal number does not refer to a legitimate signal (see *signal*(5)). Note that this request allows the server process to send signals to processes not owned by the same user ID.

TIOCTTY

Enable or disable all **termio** processing by a **pty**. **termio** processing is enabled if the **int** addressed by **arg** is nonzero and disabled if the **int** addressed by **arg** is zero. By

default, **termio** processing is enabled. **termio** processing refers to processing of input and output described by *termio(7)* (such as tab expansion), as well as the processing of the **ioctl()** requests described by *termio(7)*. When disabled, all input and output data is passed through the **pty** without modification. Issuing a **TIOCTTY ioctl()** request flushes all data buffered in the pseudo terminal and releases any processes blocked waiting for data. Enabling and disabling **TIOCTTY** affects the operation of the following **ioctl()** requests: **TIOCPKT**, **TIOCREMOTE**, **TIOCBREAK**, **TIOCSTOP**, **TIOCSTART**, **TIOC-TRAP**, and **TIOCMONITOR**.

When **TIOCTTY** is enabled, all **termio ioctl()** requests execute as specified in *termio(7)*, regardless of the side from which the **ioctl()** request is made. When **TIOCTTY** is disabled, master side **termio ioctl()** requests set and return the the external variable **errno** to **[EINVAL]**. Slave side **termio ioctl()** requests are processed like any other **ioctl()** request when **TIOCTTY** is disabled. In particular, slave side **termio ioctl()** requests set and return the external variable **errno** to **[EINVAL]** when both **TIOCTTY** and **TIOCTRAP** are disabled. (See the discussion of **ioctl()**, **open()**, and **close()** trapping below). **ioctl()** requests not defined by *termio(7)* are not affected by the state of **TIOCTTY**.

Data written through a pseudo terminal with **TIOCTTY** disabled is handled in a manner similar to data flowing through a pipe. A write request blocks in the **pty** until all data has been written into the **pty**. A read request blocks if there is no data available unless the **O_NDELAY** flag is set (see *fcntl(2)*). When data is available to be read, the read request returns whatever is available, and does not wait for the number of bytes requested to be satisfied. The number of bytes a **pty** can contain in its internal memory is implementation dependent, but is at least 256 bytes in each direction. For example, a write on the slave side of a **pty** of 1024 bytes might be read on the master side by four read requests returning 256 bytes each. The size of the chunks of data that are read is not guaranteed to be consistent, but no data is lost.

The following **ioctl()** requests, defined in **<sys/ptyio.h>**, apply only to the master side of a **pty**. In particular, these **ioctl()** requests enable/disable specific modes of **pty** driver operation. These **ioctl()** requests work in series with **TIOCTTY**; that is, the mode must be enabled by its **ioctl()** request and **TIOCTTY** must be enabled for the mode to operate. The mode can be enabled or disabled regardless of the state of **TIOCTTY**.

TIOCPKT Enable or disable packet mode. Packet mode is enabled if the **int** addressed by *arg* is nonzero and disabled if the **int** addressed by *arg* is zero. By default, packet mode is disabled. When applied to the master side of a pseudo terminal, each subsequent **read()** from the master side returns data written on the slave part of the pseudo terminal preceded by a zero byte (symbolically defined as **TIOCPKT_DATA**), or a single byte reflecting control status information. The value of such a status byte is composed of zero or more bit flags:

TIOCPKT_FLUSHREAD

The read queue for the slave side has been flushed.

TIOCPKT_FLUSHWRITE

The write queue for the slave side has been flushed.

TIOCPKT_STOP

Data flowing from the slave side of the **pty** to the master side has been stopped by means of **^S**, **TIOCSTOP**, or **TCXONC**.

TIOCPKT_START

Data flowing from the slave side of the **pty** to the master side has been restarted.

TIOCPKT_DOSTOP

Stop and start characters have been set to **^S** or **^Q**.

TIOCPKT_NOSTOP

Stop and start characters are set to something other than **^S** or **^Q**.

TIOCREMOTE Enable or disable remote mode. Remote mode is enabled if the **int** value of *arg* is nonzero and disabled if the **int** value of *arg* is zero. By default, remote mode is disabled. Remote mode is independent of packet mode. This mode causes input to the pseudo terminal to be flow controlled and not input edited (regardless of the terminal mode). Each write to the master side produces a record boundary for the process reading the slave side. In normal

usage, writing data is like typing the data as a line on a terminal; writing zero bytes is equivalent to typing an end-of-file character (that is, the EOF character as defined in *termio(7)*). The data read by the slave side is identical to the data written on the master side. Data written on the slave side and read on the master side with **TIOCREMOTE** enabled is still subject to the normal *termio(7)* processing. **TIOCREMOTE** can be used when doing remote line editing in a window manager, or whenever flow-controlled input is required. Issuing a **TIOCMONITOR** **ioctl()** request flushes all data buffered in the pseudo terminal.

The following **ioctl()** requests, defined in `<sys/ptyio.h>`, apply only to the master side of **pty**. In particular, these **ioctl()** requests are only recognized when **TIOCTTY** is enabled. When **TIOCTTY** is disabled, these **ioctl()** requests set and return the external variable **errno** to [EINVAL].

TIOCBREAK Cause a break operation to be done on the slave side of the **pty**, as if a user had pressed the break key on a real terminal. Takes no parameter.

TIOCSTOP Stop data flowing from the slave side of the **pty** to the master side (equivalent to typing **^S**). Takes no parameter.

TIOCSTART Restart output (stopped by **TIOCSTOP** or by typing **^S**). Takes no parameter.

Flow-Control Input and Output Processing

The following terms are used to describe the flow of data through pseudo terminals. INPUT refers to data flowing from the master side of a **pty** to the slave side. OUTPUT refers to data flowing from the slave side of a **pty** to the master side.

When packet mode (**TIOCPKT**) is disabled and INPUT is stopped (see **IXOFF**, input modes, in *termio(7)*), the next **read()** from the master side of a **pty** returns a STOP character. When INPUT is restarted, the next **read()** from the master side returns a START character. If packet mode (**TIOCPKT**) is enabled, the STOP or START character is preceded by a data packet indicator (**TIOCPKTDATA**). **select()** should be used by the master-side server before each **write()** request to properly handle INPUT flow control (see *select(2)*).

When INPUT flow control is enabled, **write()** and **select()** are handled as follows: Write-selects on the master side of a **pty** return true only if INPUT has not been stopped. If INPUT becomes stopped while data is being written into the master side of a **pty**, the write returns with the number of bytes written before INPUT was stopped. Writes done after INPUT is stopped return immediately with zero bytes written.

When packet mode (**TIOCPKT**) is disabled and OUTPUT is stopped (see **IXON**, input modes in *termio(7)*), each subsequent **read()** from the master side of a **pty** returns with no data read. When OUTPUT is restarted, each subsequent **read()** from the master side returns data written on the slave side. If packet mode (**TIOCPKT**) is enabled, the first **read()** after OUTPUT has been stopped returns a **TIOCPKTSTOP** packet. All subsequent reads from the master side while OUTPUT is stopped returns a **TIOCPKTDATA** packet with no data. When OUTPUT is restarted, the next **read()** from the master side returns a **TIOCPKTSTART** packet. All subsequent reads from the master side return data written on the slave side preceded by a **TIOCPKTDATA** packet. **select()** should be used by the master-side server before each **read()** to properly handle OUTPUT flow control. Otherwise, reads from the master side of a **pty** will not be prevented when OUTPUT is stopped.

Trapping **ioctl()**, **open()**, **close()** Requests

When trapping is enabled, the master side is notified when the application on its slave side makes an **ioctl()**, **open()**, or **close()** request. For trapped **ioctl()** and **open()** requests, the slave side is blocked (that is, the request does not complete) until the server on its master side acknowledges the trapped request. For trapped **close()** requests, the slave side does not block for an acknowledgement.

select() should be used by the master side server to receive notification of trapped **ioctl()**, **open()**, and **close()** requests. When one of these requests is trapped, the **select()** returns with an "exceptional condition" indicated for the slave side's file descriptor. Other mechanisms for receiving notification of trapped requests are defined below, but these mechanisms should be used only if **select()** is not available.

When trapping is disabled (default condition), unrecognized slave **ioctl()** requests return an error, with the external variable **errno** set to [EINVAL]. The only **ioctl()** requests recognized by the slave side are those defined by *termio(7)* and only when **TIOCTTY** is enabled. When **TIOCTTY** is disabled, no **ioctl()** requests are recognized by the slave side. If trapping is enabled and the master side closes, trapping is disabled. If the master closes during the middle of a handshake with the slave, the handshake

is done automatically.

Trapping occurs in two forms that are identified by the `ioctl()` requests that enable or disable them — `TIOCTRAP` and `TIOCMONITOR`. These two forms are distinguished by the types of requests they affect and by the capabilities they provide. Trapping `open()` and `close()` requests is enabled or disabled by `TIOCTRAP`. Trapping `ioctl()` requests not defined by `termio(7)` are enabled or disabled by `TIOCTRAP`. Trapping `ioctl()` requests defined by `termio(7)` are enabled or disabled by `TIOCTRAP` only when `TIOCTTY` is also disabled. When `TIOCTTY` is enabled, trapping `ioctl()` requests defined by `termio(7)` are enabled or disabled by `TIOCMONITOR`. Briefly, both `TIOCTRAP` and `TIOCMONITOR` trapping allow the server on the master side to examine the request's parameters, the pid making the request, etc. In addition, `TIOCTRAP` trapping allows the server to modify the parameters and return values of an `ioctl()` request.

The following `ioctl()` calls apply only to the master side of a `pty` and pertain to trapping `ioctl()`, `open()`, and `close()` requests. They are defined in `<sys/ptyio.h>`:

TIOCTRAP Enable or disable trapping of `ioctl()`, `open()`, and `close()` requests made by the application on the slave side of a `pty`. Trapping is enabled if the `int` addressed by `arg` is nonzero and disabled if the `int` addressed by `arg` is zero. By default, `TIOCTRAP` trapping is disabled.

TIOCTRAPSTATUS

Check for a pending `ioctl()`, `open()`, or `close()` trap. The argument points to an `int` that is set to one if a trap is pending and to zero if nothing is pending. Use `TIOCTRAPSTATUS` when the preferred method of a `select()` "exceptional condition" is not available.

TIOCREQCHECK

Return the trapped `ioctl()`, `open()`, or `close()` information to the master side. Use `TIOCREQCHECK` in response to either a `select()` "exceptional condition" or a `TIOCTRAPSTATUS` indicating that a trap is pending. A `TIOCREQCHECK` reads the pending `ioctl()`, `open()`, or `close()` information into the memory pointed to by the `arg` of `TIOCREQCHECK`. The information takes the form of the following `request_info` structure, defined in `<sys/ptyio.h>`:

```
struct request_info {
    int request;
    int argget;
    int argset;
    short pgrp;
    short pid;
    int errno_error;
    int return_value;
};
```

All elements of `request_info` refer to the slave side of the `pty` and include the following:

request	The <code>ioctl()</code> command received.
argget	The <code>ioctl()</code> request applied to master side to receive the trapped <code>ioctl()</code> structure, if one exists (a zero value means there is none). (When nonzero, <code>argget</code> is a <code>TIOCARGGET</code> request with the size field precomputed.)
argset	The <code>ioctl()</code> request applied to master side to send back the resulting <code>ioctl()</code> structure, if one exists (a zero value means there is none). (When nonzero, <code>argset</code> is a <code>TIOCARGSET</code> request with the size field precomputed.)
pgrp	The process group number of the process doing the operation.
pid	The process ID of the process doing the operation.
errno_error	The <code>errno</code> external variable error code (initialized to zero) returned by <code>ioctl()</code> on the slave side. When open error mode is enabled,

errno_error can be used to return an error for trapped slave **pty open()** requests. See the discussion of the **TIOCSMODES ioctl()** for further information on open error mode.

return_value

The success value (initialized to zero) returned by **ioctl()** on the slave side when **errno_error** is not set.

When the **ioctl()** argument received on the slave side is not a pointer, its value is stored as four bytes retrievable with an **ioctl()** request to the master side equal to **argget**.

When an **open()** or **close()** is being passed, **request** is set to **TIOCOPEN** or **TIOCCLOSE**, respectively. For **TIOCOPEN** and **TIOCCLOSE**, both **argget** and **argset** are zero because there is no **ioctl()** structure. When **TIOCTTY** is enabled, the *termio(7)* definition of open/close is executed first before being passed to the master side. Note that while all opens are trapped, only the last close on a particular inode for a **pty** slave side is trapped by the **pty**.

A **TIOCREQCHECK** returns the external variable **errno** error [EINVAL] if no **ioctl()**, **open()**, or **close()** trap is pending. Accordingly, a **TIOCREQCHECK** that returns [EINVAL] in response to a **select()** "exceptional condition" indicates that the trapped **ioctl()**, **open()**, or **close()** request was terminated by a signal after **select()** returned.

TIOCREQGET Identical to **TIOCREQCHECK** except when no **ioctl()**, **open()**, or **close()** trap is pending. A **TIOCREQGET** blocks until a slave side **ioctl()**, **open()**, or **close()** is trapped; whereas a **TIOCREQCHECK** returns [EINVAL]. Use **TIOCREQGET** when neither the preferred method of a **select()** "exceptional condition" nor the master side **ioctl()** **TIOCTRAPSTATUS** is available.

TIOCREQSET Complete the handshake started by a previous **TIOCREQCHECK** or **TIOCREQGET**. The argument should point to the **request_info** structure, as defined by the **TIOCREQCHECK**.

Before doing this **ioctl()** request to complete the handshake, the server should set **errno_error** to an external variable **errno** error value to be passed back to the slave side. If there is no error, **errno_error** can be left alone because the **pty** initializes it to zero. Also, when there is no error, **return_value** should be set if other than a zero result is desired. The server can set **return_value** and **errno_error** if the trapped request is an **ioctl()** and may set **errno_error** for a trapped **open()** if open error mode is enabled. Setting either **return_value** or **errno_error** for a trapped **close()** affects neither the return value of the request nor the external variable **errno** value of the slave side. Setting either **return_value** or **errno_error** for a trapped **open()** affects neither the return value of the request nor the external variable **errno** value of the slave side unless open error mode is enabled. Open error mode allows the server to return an error to a trapped slave **open()** by setting **errno_error**. Unlike **ioctl()** requests, setting **return_value** never affects slave **pty open()** requests. Further, setting either **return_value** or **errno_error** does not cause **TIOCREQSET** to return an error to the server.

If the **TIOCREQSET** request is made and the request value in the passed **request_info** structure does not equal the trapped value, the external variable **errno** is set and returned as [EINVAL]. [EINVAL] is also returned if there are no trapped **ioctl()**, **open()**, or **close()** requests. If the trapped request has been interrupted by a signal between the time that the server has done the **TIOCREQGET** and the **TIOCREQSET**, the **TIOCREQSET** request returns [EINVAL].

TIOCGFLAGS Get the file status flags associated with a trapped request. Upon successful return, the **ioctl()** returns in an integer referenced by *arg* the file status flags for the trapped request. The flag definitions in *<sys/file.h>* can be used to interpret the flags. If no trap is currently pending, the **TIOCGFLAGS ioctl()** returns an error with the external variable **errno** set to [EINVAL].

TIOCMONITOR

Enable or disable read-only trapping of *termio* **ioctl()** requests. **TIOCMONITOR**

trapping is enabled if the `int` addressed by `arg` is nonzero and disabled if the `int` addressed by `arg` is zero. By default, **TIOCMONITOR** trapping is disabled. **TIOCMONITOR** works in series with **TIOCTTY**; that is, the **TIOCMONITOR** trapping must be enabled and **TIOCTTY** must be enabled for **termio** `ioctl()` requests to be trapped by **TIOCMONITOR**. **TIOCMONITOR** trapping can be enabled or disabled regardless of the state of **TIOCTTY**.

When **TIOCTTY** is disabled, **termio** `ioctl()` requests are not trapped by **TIOCMONITOR**. However, `ioctl()` requests are trapped by **TIOCTRAP** if **TIOCTTY** is disabled and **TIOCTRAP** is enabled. **TIOCTRAP** trapping allows the master side server to modify the parameters and return values of an `ioctl()` request, whereas **TIOCMONITOR** trapping does not.

TIOCMONITOR trapping allows the server on the master side to know when characteristics of the line discipline in the **pty** are changed by an application on its slave side. The mechanism for handshaking **termio** requests trapped by **TIOCMONITOR** is the same as the mechanism described above for requests trapped by **TIOCTRAP**. (It is recommended that **termio** `ioctl()` requests be used on the master side to interrogate the configured state of the line discipline in the **pty**. This compensates for the window of time before **TIOCMONITOR** is enabled, when **termio** `ioctl()` requests are not trapped.)

When using `select()` on the master side of a **pty**, the "exceptional condition" refers to an `open()`, `close()`, or `ioctl()` request pending on the slave side, while "ready for reading or writing" indicates that the device can be read from or written to successfully.

Of the `ioctl()` requests subject to being trapped, only one-per-**pty** can be handled at a time. This means that when an application does a non-**termio** `ioctl()` request to the slave side, all other `ioctl()` requests to the same **pty** slave side are blocked until the first one is handshaked back by the master side. (`ioctl()` requests that are not trapped, such as **termio** when **TIOCTTY** is enabled and **TIOCMONITOR** is disabled, are not blocked.) This permits the implementation of indivisible operations by an `ioctl()` call on the slave side that is passed to the server process.

In summary, the following method of handling trapped `ioctl()`, `open()`, and `close()` requests is preferred:

1. Call `select()`. This system call blocks the master side until a slave side `ioctl()`, `open()`, or `close()` request is trapped.
2. Make **TIOCREQCHECK** `ioctl()` request. This step returns information about a trapped `ioctl()`, `open()`, or `close()` request. If **TIOCREQCHECK** returns the external variable **errno** error [EINVAL], loop back to the `select()` call.
3. Make **argget** `ioctl()` request. This optional step is used if **argget** is nonzero and the server wants to do more than just reject the trapped slave `ioctl()` request.
4. Make **argset** `ioctl()` request. This optional step is done if **argset** is nonzero and the server wants to pass back a modified `ioctl()` structure. It is done after the trapped `ioctl()` request is processed via the server on the master side.
5. Set **errno_error** and **return_value**. If the trapped request is an `ioctl()`, set **errno_error** appropriately. If the appropriate value for **errno_error** is zero, **return_value** must be set. If open error mode is enabled, set **errno_error** to a nonzero value to return an error to a trapped `open()` request.
6. Make **TIOCREQSET** `ioctl()` request. This step completes the trapped `ioctl()`, `open()`, or `close()` request.

While a process is waiting in the slave side of the **pty** for the server to complete a handshake, it is susceptible to receiving signals. The following master side `ioctl()` request allows the server process to control how the **pty** responds when a signal attempts to interrupt a trapped `open()` or `ioctl()` request:

TIOCSIGMODE

Set the signal handling state of the **pty** to the mode specified as the argument. The mode can have three values, which are **TIOCSIGBLOCK**, **TIOCSIGABORT**, and **TIOCSIGNORMAL**.

TIOCSIGBLOCK

Cause some signals to be postponed that are destined for the slave-side process whose `open()` or `ioctl()` request is trapped. Signals are postponed if they would

otherwise cause the process to jump to an installed signal handler. Signals are not postponed if they would otherwise cause the process to abort or if they are being ignored. When the server process completes the handshake by means of the **TIOCREQSET ioctl()** request, the process returns to the calling program and any pending signals are then acted upon. Any signals that the user has blocked by means of **sigblock()** continues to be blocked.

TIOCSIGABORT

Prevent a trapped **open()** or **ioctl()** request from being restarted. The server process sets this mode when it wants the interrupted requests to return to the calling program with an [EINTR] error.

TIOCSIGNORMAL

This is the default mode of the **pty**. If a signal interrupts a trapped **open()** or **ioctl()** request, the user's signal handler routine can specify whether the request is to be restarted. If the request is restarted, it executes again from the beginning and the server has to make another **TIOCREQGET** request to start the handshake over again. If the user's signal handler routine specifies that the interrupted request should not be restarted, the request returns to the calling program with [EINTR] upon completion of the signal handler. Note that the restarted request is not necessarily the very next one to be trapped.

The following **ioctl()** requests, defined in **<sys/ptyio.h>**, provide a mechanism to get and set **pty** modes. Five of the modes can also be manipulated using other **ioctl()** requests discussed previously. See the bit definitions for the **ioctl()** equivalents. The effect of enabling or disabling them by either means is identical. Commonly, an application would use the **TIOCGMODES ioctl()** to get the **pty** modes currently in effect, set or clear the bits for the modes being changed, and issue a **TIOCGMODES ioctl()** to effect the desired change.

TIOCGMODES Get the **pty** modes currently in effect. The **ioctl()** returns in a long referenced by *arg* bits indicating the states of various **pty** modes. If a bit is set, the associated mode is enabled. If a bit is clear, the associated mode is disabled. Unused bits are clear. The meaning of the bits is described under the description of the **TIOCSMODES ioctl()**.

TIOCSMODES Set the **pty** modes according to the value of type long referenced by *arg*. Unused bits are ignored but should be set to zero. The bit values for **pty** modes are listed below.

PM_REMOTE

Enable or disable remote mode. See the discussion of the **TIOCREMOTE ioctl()**.

PM_TTY

Enable or disable tty mode. See the discussion of the **TIOCTTY ioctl()**.

PM_PKT

Enable or disable packet mode. See the discussion of the **TIOCPKT ioctl()**.

PM_TRAP

Enable or disable trap mode. See the discussion of the **TIOCTRAP ioctl()**.

PM_MONITOR

Enable or disable monitor mode. See the discussion of the **TIOCMONITOR ioctl()**.

PM_OPEN_ERROR

Enable or disable open error mode. Open error mode allows a server process to return an error to a trapped slave **pty open()** through the **TIOCREQSET ioctl()**. When open error mode is enabled, the server may return a trapped **open()** with an error by setting the **errno_error** field in the **request_info** structure passed to the **TIOCREQSET ioctl()**. When open error mode is disabled (the default state), setting **errno_error** to handshake a slave **open()** has no effect. Note that unlike the **ioctl()** trap handshaking, setting **return_value** has no effect for a slave **open()** regardless of the state of open error mode. See the discussion of the **TIOCREQSET ioctl()** for further details on handshaking a trapped request.

WARNINGS

The slave side cannot indicate an end-of-file condition to the master side.

When using **TIOCREMOTE**, a single **write()** request to the master side of greater than 256 bytes may result in multiple smaller records being read from the slave side instead of only one record.

AUTHOR

pty was developed by the University of California, Berkeley.

FILES

<code>/dev/ptym/pty[a-ce-z][0-9][0-9]</code>	master pseudo terminals
<code>/dev/ptym/pty[a-ce-z][0-9][0-9][0-9]</code>	master pseudo terminals
<code>/dev/ptym/pty[a-ce-z][0-9a-f]</code>	master pseudo terminals
<code>/dev/pty[pqr][0-9a-f]</code>	master pseudo terminals
<code>/dev/pty/tty[a-ce-z][0-9][0-9]</code>	slave pseudo terminals
<code>/dev/pty/tty[a-ce-z][0-9][0-9][0-9]</code>	slave pseudo terminals
<code>/dev/pty/tty[a-ce-z][0-9a-f]</code>	slave pseudo terminals
<code>/dev/tty[pqr][0-9a-f]</code>	slave pseudo terminals

SEE ALSO

`close(2)`, `fcntl(2)`, `ioctl(2)`, `open(2)`, `read(2)`, `select(2)`, `sigblock(2)`, `write(2)`, `ptsname(3C)`, `signal(5)`, `termio(7)`.

NAME

routing - system support for local network packet routing

DESCRIPTION

The network facilities for HP-UX provide general packet routing support. Routing table maintenance is handled by application processes.

A routing table consists of a set of data structures used by the network facilities to select the appropriate remote host or gateway when transmitting packets. The table contains a single entry for each route to a specific network or host, as displayed by the **netstat** command with the **-r** or **-rn** options (see *netstat(1)*). Routes that are not valid are not displayed.

```
# netstat -r
Routing tables
Destination      Gateway          Flags  Refs  Use Interface  Pmtu
hpindwr.cup.hp.com
  localhost       localhost       UH      1    39 lo0         4608
localhost        localhost       UH      0    68 lo0         4608
147.253.56.195    localhost       UH      0     0 lo0         4608
147.253.144.66    localhost       UH      0     0 lo0         4608
default          hpinsmh.cup.hp.com
                  UG      1    21 lan0        1500
15.13.136        hpindwr.cup.hp.com
                  U      1    92 lan0        1500
147.253.56       147.253.56.195  U      0     7 lan2        1500
147.253.144.64   147.253.144.66  U      0     7 lan1        1500
```

```
# netstat -rn
Routing tables
Destination      Gateway          Flags  Refs  Use Interface  Pmtu
15.13.136.66     127.0.0.1       UH      1    39 lo0         4608
127.0.0.1        127.0.0.1       UH      0    68 lo0         4608
147.253.56.195    127.0.0.1       UH      0     0 lo0         4608
147.253.144.66    127.0.0.1       UH      0     0 lo0         4608
default          15.13.136.11    UG      2    30 lan0        1500
15.13.136.0       15.13.136.66    U      1   113 lan0        1500
147.253.56.0      147.253.56.195  U      0     7 lan2        1500
147.253.144.64    147.253.144.66  U      0     7 lan1        1500
```

```
# netstat -rv
Routing tables
Dest/Netmask     Gateway          Flags  Refs  Use Interface  Pmtu
hpindwr.cup.hp.com/0xffffffff
  localhost       localhost       UH      1    39 lo0         4608
localhost/0xffffffff
  localhost       localhost       UH      0    68 lo0         4608
147.253.56.195/0xffffffff
  localhost       localhost       UH      0     0 lo0         4608
147.253.144.66/0xffffffff
  localhost       localhost       UH      0     0 lo0         4608
default/0x00000000
  hpinsmh.cup.hp.com
                  UG      2    31 lan0        1500
15.13.136/0xfffff800
  hpindwr.cup.hp.com
                  U      1   129 lan0        1500
147.253.56/0xfffffe00
  147.253.56.195  U      0     7 lan2        1500
147.253.144.64/0xfffffff0
  147.253.144.66  U      0     7 lan1        1500
```

```
# netstat -rnv
Routing tables
Dest/Netmask      Gateway          Flags  Refs  Use Interface Pmtu
15.13.136.66/255.255.255.255
    127.0.0.1      UH              1     39 lo0         4608
127.0.0.1/255.255.255.255
    127.0.0.1      UH              0     68 lo0         4608
147.253.56.195/255.255.255.255
    127.0.0.1      UH              0      0 lo0         4608
147.253.144.66/255.255.255.255
    127.0.0.1      UH              0      0 lo0         4608
default/0.0.0.0  15.13.136.11    UG       3     40 lan0        1500
15.13.136.0/255.255.248.0
    15.13.136.66   U               1    153 lan0        1500
147.253.56.0/255.255.254.0
    147.253.56.195 U               0      8 lan2        1500
147.253.144.64/255.255.255.240
    147.253.144.66 U               0      8 lan1        1500
```

The following columns are of particular interest:

Destination	The destination Internet address: host name, network name, or default . The default keyword indicates a wildcard route, used as a last resort if no route is specified for a particular remote host or network. See Flags .
Netmask	The netmask and the destination Internet address together define a range of IP addresses that may be reached by the route's gateway. A host route by default has a netmask of all 1's. A default route by default has a netmask of all 0's. The netmask is also used in selecting a route to forward an IP packet. (see Routing Algorithm .)
Gateway	The gateway to use to get to the destination: remote gateway or the local host. See Flags .
Flags	The type of route: <ul style="list-style-type: none"> U The route is "up" or available (see <i>ifconfig(1M)</i>). G The route uses a remote host as a gateway; otherwise, the local host is shown as the gateway (see <i>route(1M)</i>). H The destination is a host; otherwise, the destination is a network (see <i>route(1M)</i>).
Interface	The interface connections: <ul style="list-style-type: none"> lo0 The local loopback after system boot-up. lan0, lan1,... The interface cards installed on the local host after the ifconfig command is executed at boot time (see <i>ifconfig(1M)</i>).

The values of the *count* and *destination* type fields in the **route** command determine the presence of the **G** and **H** flags in the **netstat -r** display and thus the route type, as shown in the following table.

Count	Destination Type	Flags	Route Type
=0	network	U	Route to a network directly from the local host
>0	network	UG	Route to a network through a remote host gateway
=0	host	UH	Route to a remote host directly from the local host
>0	host	UGH	Route to a remote host through a remote host gateway
=0	default	U	Wildcard route directly from the local host
>0	default	UG	Wildcard route through a remote host gateway

Subnets

The network facilities support variable-length subnetting. An Internet address is made up of a **network address** portion, and a **host address** portion of an address of the form:

```
192.34.17.0
```

Subnet addresses are defined as a portion of the network's Internet address. This scheme provides for:

- Network addresses that identify physically distinct networks.
- Subnet addresses that identify physically distinct subnetworks of the same network.

A network manager can subdivide the Internet address of the local network into subnets using the host number space. This facility allows several physical networks to share a single Internet address.

To allow for this, three Internet classes are defined, each accommodating a different amount of network and host addresses. The address classes are defined by the most significant bit of the binary form of the address.

The following table lists the number of networks, nodes, and the address ranges for each address class:

Class	Networks	Nodes per Network	Address Range
A	127	16777215	0.0.0.1 - 127.225.225.254
B	16383	65535	128.0.0.1 - 191.255.255.254
C	2097151	255	192.0.0.1 - 223.244.244.243
Reserved	—	—	224.0.0.0 - 255.255.255.255

The first 8 bits of a Class A network has network space for only 127, while accommodating the largest number of nodes possible among the classes defined. A single class B network has the network address limitation of 16 bits, and 16 bits to define the nodes.

For example, a Class C address space is as follows:

Indicates Class C networks	Class C subnet portion
10000000.00000110.00000001.11100001	
Network Address = 192.6.1	Host Address = 1

A subnet for a given host is specified with the **ifconfig** command (see *ifconfig(1M)*), using the **netmask** parameter with a 32-bit subnet *mask*.

The default masks for the three classes of Internet addresses are as follows:

Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0

An example Class C network number is 192.34.17.0. The last field specifies the host number. Thus, all hosts with the prefix 192.34.17 are recognized as being on the same logical and physical network.

If subnets are not in use, the default mask used is 255.255.255.0.

If subnets are used and the 8-bit host field is partitioned into 3 bits of subnet and 5 bits of host as in the above example, then the subnet mask would be 255.255.255.192.

If a host has multiple interfaces, then it can belong to different subnets. Unlike the past releases, the subnets can have different sizes even if they may have the same network address. This is accomplished by using a different netmask on each of the host interfaces. For example, the lan1 and lan2 interface shown in the netstat tables above are connected to two distinct subnets of the same network, 147.253. The subnet that lan1 belongs to can have at most 14 hosts, because its netmask is 255.255.255.240. Note, the host portion of those IP addresses in the subnet cannot be all 1's or all 0's, therefore this subnet cannot support 16 hosts, but only 14 hosts instead. The subnet that lan2 belongs to can have up to 510 hosts, because its netmask is 255.255.254.0.

Supernets

A supernet is a collection of smaller networks. Supernetting is a technique of using the netmask to aggregate a collection of smaller networks into a supernet. This technique is particularly useful for class C networks. A Class C network can only have 254 hosts. This can be too restrictive for some companies. For these companies, a netmask that only contains a portion of the network part can be applied to the hosts in

these class C networks to form a supernet. This supernet netmask should be applied to those interfaces that connect to the supernet using the *ifconfig* command (see *ifconfig*(1M)). For example, a host can configure its interface to connect to a class C supernet, 192.6, by configuring an IP address of 192.6.1.1 and a netmask of 255.255.0.0 to its interface.

Routing Algorithm

The routing table entries are of three types:

- Entries for a specific host.
- Entries for all hosts on a specific network.
- Wildcard entries for any destination not matched by entries of the first two types.

To select a route for forwarding an IP packet, the network facilities select the complete set of "matching" routing table entries from the routing table. A routing table entry is considered a match, if the result of the bit-wise AND operation between the netmask in the routing entry and the IP packet's destination address equals to the destination address in the routing entry.

The network facilities then select from the set the routing entries that have the longest netmask. The length of a netmask is defined as the number of contiguous 1 bits starting from the leftmost bit position in the 32-bit netmask field. In other words, the network facilities select the routing entry that specifies the narrowest range of IP addresses. For example, the host route entry that has a destination/netmask pair of (147.253.56.1, 0xffffffff), is more specific than the network route entry that has a destination/netmask pair of (147.253.56.0, 0xfffffe00), therefore the network facilities select the host route entry. The default route by default has a destination/netmask pair of (0.0). Therefore the default route matches all destinations but it is also the least specific. The default route will be selected only if there is not a more specific route.

There may still be multiple routing entries remaining. In that case the IP packet is routed over the first entry displayed by **netstat -r**. Such multiple routes include:

- Two or more routes to a host via different gateways.
- Two or more routes to a network via different gateways.
- Two default routes.

A superuser can change entries in the table by using the **route** command (see *route*(1M)), or by information received in Internet Control Message Protocol (ICMP) redirect messages.

WARNINGS

Reciprocal **route** commands must be executed on the local host and the destination host, as well as all intermediate hosts, if routing is to succeed in the cases of virtual circuit connections or bidirectional datagram transfers.

AUTHOR

routing was developed by the University of California, Berkeley.

FILES

/etc/hosts
/etc/networks

SEE ALSO

netstat(1), ifconfig(1M), route(1M).

NAME

sad - STREAMS Administrative Driver

SYNOPSIS

```
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/sad.h>
#include <stropts.h>

int ioctl(int fildes, int command, ... /* arg */);
```

DESCRIPTION

The **sad** driver provides an interface to the **autopush** facility using the **ioctl()** function. As an interface, the **sad** driver enables administrative tasks to be performed on STREAMS modules and drivers. By specifying the *command* parameter to the **ioctl()** function, an administrator can configure **auto-push** information for a device, get information on a device, or check a list of modules.

fildes is a file descriptor obtained by opening **/dev/sad** using **open()**. *command* specifies the administrative function to be performed. *arg* points to a data structure. If *command* is **SAD_SAP** or **SAD_GAP**, *arg* points to a struct of type **strapush**. If *command* is **SAD_VML**, *arg* points to a struct of type **str_list**.

ioctl Commands

The commands used to perform administrative functions on a STREAMS module or driver are specified by the following **ioctl()** commands:

SAD_SAP Allows the superuser to configure **autopush** information for a device. The *arg* parameter points to a **strapush** structure (defined in the **<sys/sad.h>** header file), whose members are as follows:

```
struct strapush {
    uint sap_cmd;
    long sap_major;
    long sap_minor;
    long sap_lastminor;
    long sap_npush;
    char sap_list[MAXAPUSH][FMNAMESZ+1];
};
```

sap_cmd Allows you to specify the type of configuration to perform. This field can have the following values:

SAP_ALL	Configures all minor devices.
SAP_RANGE	Configures a range of minor devices.
SAP_ONE	Configures a single minor device.
SAP_CLEAR	Clears the previous settings. Specify only the sap_major and sap_minor fields when using this command. If a previous entry specified SAP_ALL , set the sap_minor field to 0 (zero). If a previous entry was specified as SAP_RANGE , set the sap_minor field to the lowest minor device number in the range.

sap_major Specifies the major device number.

sap_minor Specifies the minor device number.

sap_lastminor Specifies the range of minor devices.

sap_npush Specifies the number of modules to push. This number must be no more than **MAXAPUSH**, which is defined in **<sad.h>**. Additionally, this number must not exceed **NSTRPUSH**.

sap_list Specifies, in order, the array of modules to push.

SAD_GAP Lets you use the **sad** driver to obtain **autopush** configuration information for a device by setting the **sap_major** and **sap_minor** fields of the **strapush** structure (see the **SAD_SAP** command) to the major and minor device numbers of the device being queried.

arg should point to a struct of type **strapush**. Upon successful completion, the **strapush** structure contains all of the information used to configure the device. Values of 0 (zero) will appear in any unused entry in the module list.

SAD_VML Enables you to check a list of modules. For example, you can determine if a specific module has been installed. The *arg* parameter points to a **str_list** structure (defined in the **<stropts.h>** header file), whose members are as follows:

```
struct str_list {
    int sl_nmods;
    struct str_mlist *sl_modlist;
};
```

sl_nmods Specifies the number of entries you have allocated in an array.

sl_modlist Points to the array of module names. The **str_mlist** structure (also in the **<stropts.h>** header file) is as follows:

```
struct str_mlist {
    char l_name[FMNAMESZ+1];
};
```

where **l_name** specifies the array of module names.

If the **l_name** array is valid, the **SAD_VML** command returns a value of 0 (zero). If the array contains an invalid module name, the command returns a value of 1. Upon failure, the command returns a value of -1.

Notes

As a STREAMS driver, **sad** also supports the normal STREAMS **I_STR ioctl()**:

```
int ioctl(fildes, I_STR, strp);
int fildes;
struct strioctl *strp;
```

In this form, specify the **ic_cmd** field in the **strioctl** structure to either **SAD_SAP**, **SAD_GAP**, or **SAD_VML**. The **ic_dp** field points to the **strapush** structure (see the **SAD_SAP** command in the **DESCRIPTION** section). Refer to the *streamio(7)* reference page for further details.

RETURN VALUE

Unless specified otherwise, upon successful completion, the **sad ioctl()** commands return a value of 0 (zero). Otherwise, a value of -1 is returned.

ERRORS

If any of the following conditions occur, the **sad ioctl** commands return the corresponding value:

SAD_SAP

[EEXIST]	The specified major/minor device number pair (sad_major/sad_minor) has already been configured.
[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
[EINVAL]	The major device number (sad_major) is invalid, the number of modules (sap_list[MAXAPUSH][FMNAMESZ+1]) is invalid, or the list of module names is invalid.
[ENODEV]	The device is not configured for autopush . This value is returned from a SAD_GAP command.
[ENOSR]	A internal autopush data structure cannot be allocated.
[ENOSTR]	The major device does not represent a STREAMS driver.
[ERANGE]	The sap_lastminor field is less than the sap_minor field when the command is SAP_RANGE , or the minor device specified in a SAP_CLEAR

		command does not exist.
	[EACCES]	Only superuser is allowed to execute the SAD_SAP ioctl .
SAD_GAP		
	[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
	[EINVAL]	The major device number (sad_major) is invalid.
	[ENODEV]	The device is not configured for autopush .
	[ENOSTR]	The major device does not represent a STREAMS driver.
SAD_VML		
	[EFAULT]	The <i>arg</i> parameter points outside the allocated address space.
	[EINVAL]	The list of module names is invalid.

SEE ALSO

autopush(1M), ioctl(2), open(2), streamio(7).

NAME

scsi - Small Computer System Interface device drivers

DESCRIPTION

The Small Computer System Interface (SCSI) is an American National Standard for interconnecting computers and peripheral devices. HP-UX supports the SCSI device protocol on parallel SCSI interfaces (see ANSI Std X3.131-199X, "SCSI-2") and Fibre Channel interfaces (see ANSI Std X3.269-199X, "Fibre Channel Protocol for SCSI"). The SCSI standard includes specifications for a variety of device types. This section describes the general SCSI interface for all SCSI device drivers. Information about specific device types can be found in the manual sections which describe SCSI peripheral device drivers for those device types.

The **SIOC_INQUIRY** ioctl is supported by all SCSI device drivers. This ioctl returns the SCSI device-specific INQUIRY command data. This data contains device identification and capability information. Since there have been multiple versions of the SCSI standard for inquiry data, multiple versions of the inquiry data declaration are provided. The SCSI-1 version is provided for backward compatibility only.

The **SIOC_CAPACITY** ioctl indicates the current device size. A device size is defined to be a logical block size and some number of logical blocks. The means of determining this device-size data is particular to the specific device type. Logical block size and/or number of logical blocks equal to zero indicates: the device size is unknown, the device is not currently capable of I/O operations, or I/O operations are not meaningful for the device. Note that **DIOC_CAPACITY** is preferred (see *disk(7)*).

The header file `<sys/scsi.h>` has useful information for SCSI devices. The following is included from `<sys/scsi.h>`:

```
#define SIOC_INQUIRY          _IOR('S', 2, union inquiry_data)
#define SIOC_CAPACITY        _IOR('S', 3, struct capacity)

/* SCSI-1 inquiry structure */
struct inquiry {
    unsigned char    dev_type;
    unsigned int     rmb:1;
    unsigned int     dtq:7;
    unsigned int     iso:2;
    unsigned int     ecma:3;
    unsigned int     ansi:3;
    unsigned int     resv:4;
    unsigned int     rdf:4;
    unsigned char    added_len;
    unsigned char    dev_class[3];
    char             vendor_id[8];
    char             product_id[16];
    char             rev_num[4];
    unsigned char    vendor_spec[20];
    unsigned char    resv4[40];
    unsigned char    vendor_parm_bytes[32];
};

/* SCSI-2 inquiry structure */
struct inquiry_2 {
    unsigned int     periph_qualifier:3;
    unsigned int     dev_type:5;
    unsigned int     rmb:1;
    unsigned int     dtq:7;
    unsigned int     iso:2;
    unsigned int     ecma:3;
    unsigned int     ansi:3;
    unsigned int     aenc:1;
    unsigned int     trmiop:1;
    unsigned int     resv1:2;
    unsigned int     rdf:4;
    unsigned char    added_len;
    unsigned char    resv2[2];
    unsigned int     reladr:1;
    unsigned int     wbus32:1;
};
```



```

        unsigned int    wbus16:1;
        unsigned int    sync:1;
        unsigned int    linked:1;
        unsigned int    resv3:1;
        unsigned int    cmdque:1;
        unsigned int    sftre:1;
        char            vendor_id[8];
        char            product_id[16];
        char            rev_num[4];
        unsigned char    vendor_spec[20];
        unsigned char    resv4[40];
        unsigned char    vendor_parm_bytes[32];
    };

    /* union for SIOC_INQUIRY ioctl */
    union inquiry_data {
        struct inquiry    inq1;        /* SCSI-1 inquiry */
        struct inquiry_2  inq2;        /* SCSI-2 inquiry */
    };

    /* structure for SIOC_CAPACITY ioctl */
    struct capacity {
        int    lba;
        int    blkksz;
    };

```

The **SIOC_XSENSE** ioctl returns detailed information about device status and errors when such information is available. Since there have been multiple versions of the SCSI standard for sense (status) data, multiple versions of the sense data declaration are provided. The SCSI-1 and non-aligned versions are provided for backward compatibility only. If no new CHECK-CONDITION-caused REQUEST SENSE command data has been obtained since the last **SIOC_XSENSE** ioctl call, the **xsense_aligned.error_class** and **sense_2_aligned.error_code** fields will contain the value zero. Applications which require more accurate REQUEST SENSE data handling should use the SCSI device-control driver (see *scsi_ctl(7)*).

The following information is included from `<sys/scsi.h>`:

```

#define SIOC_XSENSE            _IOR('S', 7, union sense_data)

/* structure for SIOC_XSENSE ioctl */
union sense_data {
    struct xsense_aligned r_sensela; /* SCSI and CCS devices */
    struct sense_2_aligned r_sense2a; /* SCSI-2 devices */
};

/* structure for SCSI-1 and SCSI-CCS sense data */
struct xsense_aligned {
    unsigned int    valid:1;
    unsigned int    error_class:3;
    unsigned int    error_code:4;
    unsigned char    seg_num;
    unsigned int    parms:4;
    unsigned int    sense_key:4;
    unsigned char    lba[4];
    unsigned char    add_len;
    unsigned char    copysearch[4];
    unsigned char    sense_code;
    unsigned char    resv;
    unsigned char    fru;
    unsigned char    field;
    unsigned char    field_ptr[2];
    unsigned char    dev_error[4];
    unsigned char    misc_bytes[106];
};

/* structure for SCSI-2 sense data */

```

S

```

struct sense_2_aligned {
    unsigned int    info_valid:1;
    unsigned int    error_code:7;
    unsigned char   seg_num;
    unsigned int    filemark:1;
    unsigned int    eom:1;
    unsigned int    ili:1;
    unsigned int    resv:1;
    unsigned int    key:4;
    unsigned char   info[4];
    unsigned char   add_len;
    unsigned char   cmd_info[4];
    unsigned char   code;
    unsigned char   qualifier;
    unsigned char   fru;
    unsigned char   key_specific[3];
    unsigned char   add_sense_bytes[113];
};

```

ERRORS

The following errors may result from a call to a SCSI device driver:

[EACCES]	Required permission is denied for the the device or operation.
[ENXIO]	If resulting from an open call, this indicates there is no device at the specified address. For other calls, this indicates the specified address is out of range or the device may no longer be accessed.
[EINVAL]	If resulting from an open call, this indicates the device is not supported by the device driver (e.g. incorrect device type). For other calls, this indicates the request or some request argument is invalid.
[EBUSY]	This indicates the device is not ready for use or that the requested operation conflicts with other operations (e.g. the device is currently open via another device driver or exclusive access is in effect).
[EIO]	Indicates a SCSI protocol or communication problem has occurred, or that a SCSI command resulted in a non-good status.

Manual entries that describe specific SCSI peripheral device drivers may provide additional qualification of error results.

WARNINGS

Use of devices that are not officially supported can cause data loss, system panics and device damage. HP-UX device drivers expect devices to be SCSI-2 compliant. Unsupported devices that are only SCSI-CCS compliant may work but their use is discouraged. Use of unsupported devices that are only SCSI-1 compliant is strongly discouraged.

Changing SCSI bus connectivity (recabling) while the system is running is not supported. Switching SCSI device power on or off while the device is connected to a system that does not support powerfail recovery is not supported. These activities are known to cause data loss and system panics.

On systems that support the `scsi_ctl` interface, the `SIOC_CMD_MODE`, `SIOC_SET_CMD`, and `SIOC_RETURN_STATUS` ioctls are obsolete (see *scsi_ctl(7)*). Direct manipulation of SCSI devices via the `scsi_ctl` interface provides a more functionally complete and easier-to-use means of low level SCSI device control (see *scsi_ctl(7)*).

Drivers that support only devices which have no meaningful size may not support the `SIOC_CAPACITY` ioctl. Total device size in bytes may exceed $2^{32}-1$ for some devices.

DEPENDENCIES

sdisk/schgr/sflop/stape

The `SIOC_EXCLUSIVE` ioctl may be used to obtain and release exclusive access. Exclusive access, which prevents simultaneous access by other applications, is required for some operations and may be desirable in other circumstances. The following exclusive access control arguments are supported:

- 0 Release exclusive access to logical unit (LUN).
- 1 Gain exclusive access to logical unit (LUN).
- 2 Release exclusive access to associated SCSI target.
- 3 Gain exclusive access to associated SCSI target.
- 4 Release exclusive access to associated SCSI bus.
- 5 Gain exclusive access to associated SCSI bus.

The `SIOC_MEDIUM_CHANGED` ioctl indicates when the media in a removable-media device may have changed. A value of "1" indicates the device media may have changed since the last `SIOC_MEDIUM_CHANGED` ioctl call. Note that only the first such call after a media change receives this indication. This means that media changes are likely to be missed if multiple applications are attempting to detect media changes. Exclusive access, obtained through use of the `SIOC_EXCLUSIVE` ioctl, can be used to avoid this problem.

The following information is included from `<sys/scsi.h>`:

```
#define SIOC_MEDIUM_CHANGED _IOR('S', 42, int)
#define SIOC_EXCLUSIVE      _IOR('S', 68, int)
```

disc3

The `SIOC_VPD_INQUIRY` ioctl allows access to detailed device specific information. The `page_code` field specifies which SCSI vital product data page is requested. The `page_buf` field is filled with the requested page data.

The following information is included from `<sys/scsi.h>`:

```
#define SIOC_VPD_INQUIRY _IOWR('S', 10, struct vpd_inquiry)
/* union for SIOC_VPD_INQUIRY ioctl */
struct vpd_inquiry {
    char    page_code;          /* VPD page code */
    char    page_buf[126];     /* buffer for VPD page info */
};
```

FILES

`/usr/include/sys/scsi.h`

SEE ALSO

`diskinfo(1M)`, `ioctl(2)` `autochanger(7)`, `scsi_ctl(7)`, `scsi_disk(7)`, `scsi_tape(7)`.

NAME

scsi_ctl - SCSI pass-through driver

DESCRIPTION

SCSI devices are controlled by a device-specific driver, when one exists. Device-specific drivers, such as those for SCSI direct access (disk) and sequential access (tape) devices, coordinate device and driver states to accomplish correct logical device behavior. The **scsi_ctl** pass-through driver enables use of SCSI devices and commands not normally supported by these device-specific drivers.

Depending on the minor number, a successful **open()** call by **scsi_ctl** might or might not require the device exist and respond to a SCSI Inquiry command. Once open, **ioctl()** calls can be used to change SCSI communication parameters or attempt SCSI commands and other SCSI operations. Since **scsi_ctl** does not attempt to logically understand the target device, **read()** and **write()** calls are not supported.

Except where noted, the ioctls described here are available through all SCSI device drivers (including device-specific drivers). Super-user privileges or device write permissions are required to use these ioctls. All **reserved** fields in the data structures associated with these ioctls must be zero-filled.

Special Device File Minor Number

The **scsi_ctl** driver is the preferred method to perform the SIOC_IO ioctl, rather than going through a device-specific driver (such as **sdisk**). To do this, you must create the device special file for **scsi_ctl**. Use **mknod(1M)**, substituting the values in the minor number as noted:

```
/usr/sbin/mknod name c 203 0xii10o
```

where component parts of the minor number are constructed as follows:

- ii* Two hexadecimal digits, identifying the controlling interface card by its "Instance" number. The Instance value is displayed in **ioscan(1M)** output, under column **I** for the "Interface" hardware type.
- t* One hexadecimal digit identifying the drive (target) address.
- l* One hexadecimal digit identifying the logical unit number (LUN) within the device.
- 0** Hexadecimal digit zero, for reserved portion of the minor number.
- o* Optional values as follows: 0 to perform Inquiry on open to ensure the device exists (recommended); or 2 to inhibit Inquiry on open.

SCSI Communication Parameters

HP-UX supports the SCSI device protocol on parallel SCSI interfaces and Fibre Channel interfaces. The SCSI communication parameters described here might only apply to certain SCSI interfaces and are noted as such in the descriptions.

SCSI communication parameters control features related to communication for three different scope levels: bus (link), target, and logical unit number (LUN). Bus communication parameters apply to all targets connected to a specific bus. Target communication parameters apply to all LUNs associated with a specific target. LUN communication parameters apply to a specific LUN. SCSI communication parameters apply to all device drivers (both device-specific and **scsi_ctl**).

At power-up and after being reset, all parallel SCSI devices and hosts communicate using asynchronous data transfers. Asynchronous data transfers use request (REQ) and acknowledge (ACK) signaling. The strict ordering of REQ and ACK signaling simplifies the communication protocol but limits I/O performance. A SCSI target and host pair may agree to use synchronous data transfers to increase I/O performance. Synchronous data transfers improve I/O performance by lessening the ordering requirements on REQs and ACKs. By allowing multiple outstanding REQs, signal propagation delays and temporary rate imbalances are better tolerated. To make use of synchronous data transfers, a SCSI target and host must negotiate to determine mutually acceptable maximum REQ-ACK-offset and data-transfer rate parameters. The maximum REQ-ACK-offset parameter indicates the maximum allowable number of outstanding REQs. The value zero is used to indicate asynchronous data transfer. Other values indicate synchronous data transfer. The appropriate value is generally dependent on the size of the receive data FIFO. High values tend to improve data transfer rates. The maximum data-transfer rate parameter indicates the "burst" data transfer rate (minimum allowable time between successive synchronous data transfers). A SCSI synchronous data transfer request (SDTR) message, used to initiate the negotiation process, is associated with the processing of a SCSI command.

At power-up and after being reset, all parallel SCSI devices and hosts communicate using eight-bit data transfers. A SCSI target and host pair may agree to use sixteen-bit (wide) data transfers to increase I/O

performance. To make use of wide data transfers, a SCSI target and host must negotiate to determine a mutually acceptable data transfer width parameter. A SCSI wide data transfer request (WDTR) message, used to initiate the negotiation process, is associated with the processing of a SCSI command.

Some SCSI devices are able to simultaneously manage multiple active commands. Such a device has a command queue that holds commands for processing. Command queuing can improve I/O performance by reducing the time spent by the device waiting for new commands from the host. Note that command queuing might not improve I/O performance substantially for devices that support “read-ahead” and “immediate-reporting” (see *scsi_disk(7)* and *scsi_tape(7)*). The SCSI device and host use command tags to correctly manage these multiple simultaneously active commands. At all times when command queuing is in effect, each active command being handled by a specific LUN has a unique command tag.

SCSI devices indicate their ability to support the special communication features described above in their SCSI INQUIRY command data. Normally the SCSI INQUIRY command data and negotiation protocols allow hosts and devices to determine the optimal communication parameters so that I/O performance is maximized. The current operating communication parameters may be determined by use of the: **SIOC_GET_LUN_PARMS**, **SIOC_GET_TGT_PARMS**, and **SIOC_GET_BUS_PARMS** ioctls.

Occasionally, it is desirable to limit SCSI communication parameters to work around a communication problem or to provide external insight in determining optimal parameters. SCSI communication parameter limit suggestions can be specified by use of the: **SIOC_SET_LUN_LIMITS**, **SIOC_SET_TGT_LIMITS**, and **SIOC_SET_BUS_LIMITS** ioctls. Note that there might be substantial differences between specified communication parameter limit suggestions and the corresponding actual current communication parameters being used for communication. These differences are a result of device-specific driver capabilities, interface driver capabilities, interface hardware capabilities, device capabilities, delays due to the negotiation process, delays due to currently active commands, and delays due to commands waiting to be sent to devices. Note that communication parameter limit suggestions might not survive between **close()** and **open()** calls, when no SCSI device drivers (device-specific or **scsi_ctl**) have associated LUN(s) open.

The current SCSI communication parameter limit suggestions may be determined by use of the **SIOC_GET_LUN_LIMITS**, **SIOC_GET_TGT_LIMITS**, and **SIOC_GET_BUS_LIMITS** ioctls.

Logical unit communication parameters may be managed by use of the **SIOC_GET_LUN_PARMS**, **SIOC_SET_LUN_LIMITS**, and **SIOC_GET_LUN_LIMITS** ioctls.

The **SIOC_GET_LUN_PARMS** ioctl indicates the current LUN communication parameter values. The **max_q_depth** field indicates whether or not tagged queuing is enabled, and if enabled, the maximum number of simultaneously active commands allowed. When **max_q_depth** is zero, tagged queuing is disabled. When it is one, tags are being used but commands are still being serially processed. When it is greater than one, tags are being used and **max_q_depth** specifies the maximum number of simultaneously active commands allowed.

The **SIOC_SET_LUN_LIMITS** ioctl may be used to provide LUN communication parameter limit suggestions. The **max_q_depth** field indicates whether or not tagged queuing should be enabled, and if enabled, the maximum number of simultaneously active commands that should be allowed. The **SIOC_GET_LUN_LIMITS** ioctl indicates the current LUN communication parameter limit suggestions.

Target communication parameters may be managed by use of the **SIOC_GET_TGT_PARMS**, **SIOC_SET_TGT_LIMITS**, and **SIOC_GET_TGT_LIMITS** ioctls to any associated LUN.

The **SIOC_GET_TGT_PARMS** ioctl indicates the current target communication parameter values. The **width**, **reqack_offset**, and **xfer_rate** fields indicate the currently negotiated data transfer parameters. When **width** is eight, narrow transfers are in effect. When it is sixteen, wide transfers are in effect. When **reqack_offset** is zero, asynchronous transfers are in effect and **xfer_rate** is meaningless. When **reqack_offset** is non-zero, synchronous transfers are in effect and the maximum “burst” data transfer rate is **xfer_rate** words per second, where the size of a word is as indicated in **width**.

The **SIOC_SET_TGT_LIMITS** ioctl specifies the target communication parameter limit suggestions. The **max_width** field specifies maximum bus width that should be used for data transfers. The **max_reqack_offset** field specifies the maximum number of outstanding REQs that should be attempted during data transfers. The **max_xfer_rate** field specifies the maximum “burst” data rate that should be allowed during synchronous data transfers. The **SIOC_GET_TGT_LIMITS** ioctl indicates the current target communication parameter limit suggestions. The **width**, **reqack_offset**, **xfer_rate**, **max_width**, **max_reqack_offset**, **max_xfer_rate** fields only apply to parallel SCSI.

Bus communication parameters may be managed by use of the **SIOC_GET_BUS_PARMS**, **SIOC_SET_BUS_LIMITS**, and **SIOC_GET_BUS_LIMITS** ioctls to any associated LUN.

The `SIOC_GET_BUS_PARMS` ioctl indicates the current bus communication parameter values. The `max_width` field indicates the maximum data transfer width that will be attempted for data transfers to any target device connected to the associated bus. The `max_reqack_offset` field indicates the maximum number of outstanding REqs that will be attempted during data transfers to any target device connected to the associated bus. The `max_xfer_rate` field indicates the maximum “burst” data transfer rate that will be attempted for data transfers to any target device connected to the associated bus.

The `SIOC_SET_BUS_LIMITS` ioctl specifies the bus communication parameter limit suggestions for targets connected to the associated bus. The `max_width` field specifies the suggested maximum data transfer width that should be attempted for data transfers to any target device connected to the associated bus. The `max_reqack_offset` field specifies the maximum number of outstanding REqs that should be attempted during data transfers to any target device connected to the associated bus. The `max_xfer_rate` field specifies the maximum synchronous “burst” data transfer rate that should be attempted for data transfers to any target device connected to the associated bus. The `SIOC_GET_BUS_LIMITS` ioctl indicates the current bus communication parameter limit suggestions. The `max_width`, `max_reqack_offset` and `max_xfer_rate` fields only apply to parallel SCSI.

The following is included from `<sys/scsi.h>`:

```
/* SCSI communication parameter ioctls */
#define SIOC_GET_LUN_PARMS      _IOR('S', 58, struct sioc_lun_parms)
#define SIOC_GET_TGT_PARMS     _IOR('S', 59, struct sioc_tgt_parms)
#define SIOC_GET_BUS_PARMS     _IOR('S', 60, struct sioc_bus_parms)
#define SIOC_GET_LUN_LIMITS    _IOR('S', 61, struct sioc_lun_limits)
#define SIOC_GET_TGT_LIMITS    _IOR('S', 62, struct sioc_tgt_limits)
#define SIOC_GET_BUS_LIMITS    _IOR('S', 63, struct sioc_bus_limits)
#define SIOC_SET_LUN_LIMITS    _IOW('S', 64, struct sioc_lun_limits)
#define SIOC_SET_TGT_LIMITS    _IOW('S', 65, struct sioc_tgt_limits)
#define SIOC_SET_BUS_LIMITS    _IOW('S', 66, struct sioc_bus_limits)

struct sioc_lun_parms {
    unsigned int flags;
    unsigned int max_q_depth;           /* maximum active I/O's */
    unsigned int reserved[4];          /* reserved for future use */
};

struct sioc_lun_limits {
    unsigned int flags;
    unsigned int max_q_depth;
    unsigned int reserved[4];          /* reserved for future use */
};

struct sioc_tgt_parms {
    unsigned int flags;
    unsigned int width;                /* bits */
    unsigned int reqack_offset;
    unsigned int xfer_rate;            /* bytes/sec */
    unsigned int reserved[4];          /* reserved for future use */
};

struct sioc_tgt_limits {
    unsigned int flags;
    unsigned int max_reqack_offset;
    unsigned int max_xfer_rate;        /* bytes/sec */
    unsigned int max_width;            /* bits */
    unsigned int reserved[4];          /* reserved for future use */
};

struct sioc_bus_parms {
    unsigned int flags;                /* reserved for future use */
    unsigned int max_width;
    unsigned int max_reqack_offset;
    unsigned int max_xfer_rate;        /* bytes/sec */
    unsigned int reserved[4];          /* reserved for future use */
};

struct sioc_bus_limits {
```

```

        unsigned int flags;                /* reserved for future use */
        unsigned int max_width;
        unsigned int max_reqack_offset;
        unsigned int max_xfer_rate;        /* bytes/sec */
        unsigned int reserved[4];          /* reserved for future use */
    };

```

SCSI Commands and Operations

The `SIOC_IO` ioctl allows an arbitrary SCSI command to be sent to a device. All details of the SCSI command protocol are handled automatically.

The following flags can be used to specify the `flags` field value:

<code>SCTL_READ</code>	Data-in phase expected if the <code>data_length</code> field is non-zero. The absence of this flag implies that a data-out phase is expected if the <code>data_length</code> field is non-zero.
<code>SCTL_INIT_SDTR</code>	synchronous data transfer request negotiations should be attempted with this command. This flag only applies to parallel SCSI.
<code>SCTL_INIT_WDTR</code>	wide data transfer request negotiations should be attempted with this command. This flag only applies to parallel SCSI.
<code>SCTL_NO_DISC</code>	discpriv bit in Identify message is not set. This flag only applies to parallel SCSI.

The `cdb` field specifies the SCSI command bytes. The number of command bytes is specified by the `cdb_length` field. These command bytes are sent to the target device during the SCSI command phase.

The address of the data area for the data phase of the SCSI command is specified by the `data` field. The `data_length` field specifies the maximum number of data bytes to be transferred. A zero-valued `data_length` indicates that no data phase should occur. Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. The caller is responsible for correctly specifying both the `data_length` field and any cdb data length values. The length may not be larger than `SCSI_MAXPHYS` and some implementations further restrict this length.

The `max_msecs` field specifies the maximum time, in milliseconds, that the device should need to complete the command. If this period of time expires without command completion, the system might attempt recovery procedures to regain the device's attention. These recovery procedures might include abort tag, abort, and device and bus reset operations. A zero value in the `max_msec` field indicates that the timeout period is infinite and the system should wait indefinitely for command completion.

When the `SIOC_IO` ioctl call returns, all command processing has been completed. Most `SIOC_IO` ioctl calls will return zero (success). The resulting detailed ioctl data should be used to evaluate "success" or "failure" from the caller's perspective. The `cdb_status` field indicates the results of the cdb command. If the `cdb_status` field indicates a `S_CHECK_CONDITION` status, the `sense_status` field indicates the results of the SCSI `REQUEST SENSE` command used to collect the associated sense data. These status fields will contain one of the following values:

<code>SCTL_INVALID_REQUEST</code>	The SCSI command request is invalid and thus not attempted.
<code>SCTL_SELECT_TIMEOUT</code>	The target device does not answer to selection by the host SCSI interface (the device does not exist or does not respond).
<code>SCTL_INCOMPLETE</code>	The device answered selection but the command is not completed (the device took too long or a communication failure occurred).
<code>S_GOOD</code>	Device successfully completed the command.
<code>S_CHECK_CONDITION</code>	Device indicated sense data is available.
<code>S_CONDITION_MET</code>	Device successfully completed the command and the requested (search or pre-fetch) operation is satisfied.
<code>S_BUSY</code>	Device indicated it is unable to accept the command because it is busy doing other operations.
<code>S_INTERMEDIATE</code>	Device successfully completed this command, which is one in a series of linked commands (not supported, see WARNINGS).

S_I_CONDITION_MET	Device indicated both S_INTERMEDIATE and S_CONDITION_MET (not supported, see WARNINGS).
S_RESV_CONFLICT	Device indicated the command conflicted with an existing reservation.
S_COMMAND_TERMINATED	Device indicated the command is terminated early by the host system.
S_QUEUE_FULL	Device indicated it is unable to accept the command because its command queue is currently full.

The **data_xfer** field indicates the number of data bytes actually transferred during the data phase of the cdb command. This field is valid only when the **cdb_status** field contains one of the following values: **S_GOOD** or **S_CHECK_CONDITION**. The **sense_xfer** field indicates the number of valid sense data bytes. This field is valid only when the **cdb_status** field contains the value **S_CHECK_CONDITION** and the **sense_status** field contains the value **S_GOOD**.

The **SIOC_ABORT** ioctl causes a SCSI **ABORT** message to be sent to the LUN. This clears all active commands to the LUN from this initiator.

The **SIOC_RESET_DEV** ioctl causes a SCSI device to be reset (including clearing all active commands). On parallel SCSI a **SIOC_RESET_DEV** ioctl causes a SCSI **BUS DEVICE RESET** message to be sent to the associated target. On Fibre Channel a **SIOC_RESET_DEV** ioctl causes a **TARGET RESET** task management function to be sent to the associated target followed by a Global Process Logout (GPRLO).

The **SIOC_RESET_BUS** ioctl causes the system to generate a SCSI bus reset condition on the associated bus. A SCSI bus reset condition causes all devices on the bus to be reset (including clearing all active commands on all devices). The **SIOC_RESET_BUS** ioctl does not apply to Fibre Channel.

Often it is necessary or useful to prohibit other SCSI commands while performing device-control operations. Normally this should be done by gaining exclusive access via the **SIOC_EXCLUSIVE** ioctl. The associated argument points to an integer with one of these values:

- 0 release exclusive access to logical unit
- 1 obtain exclusive access to logical unit
- 2 release exclusive access to target
- 3 obtain exclusive access to target
- 4 release exclusive access to bus
- 5 obtain exclusive access to bus

Occasionally exclusive access not possible (for example, diagnostic operations on a device containing a mounted file system). "Priority mode" causes all device-specific driver I/O operations (e.g. file system I/O and virtual memory page swapping) and all SCSI device driver open calls (including **scsi_ctl** open calls) to the associated LUN to block. These I/O operations and open calls are blocked for the entire duration that priority mode is in effect. While priority mode is in effect only **SIOC_IO** operations should be attempted (these operations will not be blocked). The **SIOC_PRIORITY** ioctl controls the LUN priority mode. This ioctl is only available via the device-control driver. The value "1" enables priority mode. The value zero disables priority mode.

The header file **<sys/scsi.h>** has useful information for SCSI device control. The following is included from **<sys/scsi.h>**:

```
/* SCSI device control ioctls */
#define SIOC_IO          _IOWR('S', 22, struct sctl_io)
#define SIOC_RESET_DEV  _IO('S', 16)
#define SIOC_RESET_BUS  _IO('S', 9)
#define SIOC_PRIORITY_MODE _IOW('S', 67, int)

/* Structure for SIOC_IO ioctl */
struct sctl_io
{
    unsigned    flags;
    unsigned char cdb_length;
    unsigned char cdb[16];
    void        *data;
    unsigned    data_length;
    unsigned    max_msecs;
    unsigned    data_xfer;
    unsigned    cdb_status;
    unsigned char sense[256];
}
```



```

        unsigned      sense_status;
        unsigned char  sense_xfer;
        unsigned char  reserved[64];
};

```

EXAMPLES

Assume that *fildev* is a valid file descriptor for a SCSI device. The first example attempts a SCSI INQUIRY command:

```

#include <sys/scsi.h>

struct sctl_io sctl_io;
#define MAX_LEN 255
unsigned char inquiry_data[MAX_LEN];

memset(sctl_io, 0, sizeof(sctl_io)); /* clear reserved fields */
sctl_io.flags = SCTL_READ;           /* input data expected */
sctl_io.cdb[0] = 0x12;               /* can use scsi.h CMDInquiry */
sctl_io.cdb[1] = 0x00;
sctl_io.cdb[2] = 0x00;
sctl_io.cdb[3] = 0x00;
sctl_io.cdb[4] = MAX_LEN;           /* allocation length */
sctl_io.cdb[5] = 0x00;
sctl_io.cdb_length = 6;             /* 6 byte command */
sctl_io.data = &inquiry_data[0];   /* data buffer location */
sctl_io.data_length = MAX_LEN;      /* maximum transfer length */
sctl_io.max_msecs = 10000;          /* allow 10 seconds for cmd */
if (ioctl(fildev, SIOC_IO, &sctl_io) < 0)
{
    /* request is invalid */
}

```

The following example attempts a SCSI TEST UNIT READY command and checks to see if the device is ready, not ready, or in some other state.

```

#include <sys/scsi.h>

struct sctl_io sctl_io;

memset(sctl_io, 0, sizeof(sctl_io)); /* clear reserved fields */
sctl_io.flags = 0;                   /* no data transfer expected */
sctl_io.cdb[0] = 0x00;               /* can use CMDtest_unit_ready */
sctl_io.cdb[1] = 0x00;
sctl_io.cdb[2] = 0x00;
sctl_io.cdb[3] = 0x00;
sctl_io.cdb[4] = 0x00;
sctl_io.cdb[5] = 0x00;
sctl_io.cdb_length = 6;             /* 6 byte command */
sctl_io.data = NULL;                /* no data buffer is provided */
sctl_io.data_length = 0;            /* do not transfer data */
sctl_io.max_msecs = 10000;          /* allow 10 seconds for cmd */
if (ioctl(fildev, SIOC_IO, &sctl_io) < 0)
{
    /* request is invalid */
}
else if (sctl_io.cdb_status == S_GOOD)
{
    /* device is ready */
}
else if (sctl_io.cdb_status == S_BUSY ||
         (sctl_io.cdb_status == S_CHECK_CONDITION &&
          sctl_io.sense_status == S_GOOD &&
          sctl_io.sense_xfer > 2 &&
          (sctl_io.sense[2] & 0x0F) == 2)) /* can use sense_data */
{
    /* device is not ready */
}

```

S

```

    }
    else
    {
        /* unknown state */
    }

```

WARNINGS

Incorrect use of **scsi_ctl** operations (even those attempting access to non-existent devices) can cause data loss, system panics, and device damage.

The **SIOC_EXCLUSIVE** ioctl should be used to gain exclusive access to a device prior to attempting **SIOC_IO** commands. If exclusive access is not obtained, **SIOC_IO** commands will be intermixed with device-specific driver commands, which can lead to undesirable results.

Device-specific drivers can reject inappropriate or troublesome **SIOC_IO** commands. However, since not all such operations are known and detected, care should be exercised to avoid disrupting device-specific drivers when using commands that modify internal device states.

It is very easy to cause system deadlock through incorrect use of the **SIOC_PRIORITY_MODE** ioctl. Normally it is necessary to lock the calling process into memory (see *plock(2)*) prior to enabling priority mode.

Most SCSI commands have a logical unit number (LUN) field. Parallel SCSI implementations on the HP-UX operating system select logical units via the SCSI **IDENTIFY** message. The LUN portion of the cdb should normally be set to zero, even when the LUN being accessed is not zero.

Use of linked commands is not supported.

Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. Both the **data_length** field and any cdb data length values must be correctly specified to get correct command results.

Very large (or infinite) timeout values can cause a parallel SCSI bus (potentially the entire system) to hang.

Device and/or bus reset operations can be used to regain a device's attention when a timeout expires.

Resetting a device can cause I/O errors and/or loss of cached data. This can result in loss of data and/or system panics.

Obtaining SCSI INQUIRY data by use of the **SIOC_INQUIRY** ioctl instead of by use of the **SIOC_IO** ioctl is generally preferable since SCSI implementations on the HP-UX operating system synchronize access of inquiry data during driver open calls.

Since communication parameters can be affected by device-specific driver capabilities, device-specific driver use might result in communication parameter changes.

The **SIOC_CAPACITY** ioctl is not supported by **scsi_ctl** because the meaning of capacity is device-specific.

S

FILES

```

/usr/include/sys/scsi.h
/usr/include/sys/scsi_ctl.h

```

SEE ALSO

mknod(1M), ioctl(2), scsi(7).

NAME

scsi_disk - SCSI direct access device driver (sdisk)

DESCRIPTION

This section describes the interface for access of SCSI disk, CD-ROM, and optical disk devices through the character special device driver.

SCSI direct access devices store a sequence of data blocks. Each direct access device has a specific device size consisting of a number of data blocks and a logical block size. All data blocks have the same logical block size. Since I/O operations must have a size that is an integral number of blocks, one logical block size is the smallest possible I/O quantity. The device block size can be determined through use of the `DIOC_DESCRIBE`, `DIOC_CAPACITY`, and `SIOC_CAPACITY` ioctls (see *disk(7)* and *scsi(7)*; `SIOC_CAPACITY` is not supported on `disc3`). A direct access device that is not ready for use, whether due to no media installed or another reason, is interpreted to mean the device has zero size. An `open()` call to such a device succeeds, but subsequent `read()` and `write()` calls fail.

The *ioctl(2)* manpage explains how the operations and arguments are used. Note, the *arg* used is commonly the address of the parameter cited in the particular ioctl `#define` statement. See the **EXAMPLES** section for sample code.

To improve performance, many SCSI disk devices have caches, which can be used for both read and write operations. Read cache use, called "read ahead", causes the disk drive to read data in anticipation of read requests. Read ahead is only apparent to users in the increased performance that it produces. Write cache use is called "immediate reporting". Immediate reporting increases I/O performance by reporting a completed write status before the data being written is actually committed to media. If the subsequent physical write operation does not complete successfully, data may be lost. Physical write failures due to media defects are largely eliminated by use of automatic sparing in disk drives. Power failure between immediate reporting and media commit can result in cached data being lost. However, the period of time between these events is typically relatively small, making such losses unlikely. The `SIOC_GET_IR` ioctl can be used to determine if immediate-reporting functionality is currently being used by the device. The value 1 indicates immediate reporting is enabled. The value zero indicates immediate reporting is disabled. The `SIOC_SET_IR` ioctl can be used to enable or disable immediate reporting. A zero value disables immediate reporting. The value 1 enables immediate reporting.

The `SIOC_SYNC_CACHE` ioctl can be used to force data cached in the device to media.

Most SCSI removable media disk devices support "prevent" and "allow" media-removal commands. To avoid data corruption and data accessibility problems, media removal is prevented for the entire duration a removable media disk device is open. Because media removal is not supported, the `SIOC_MEDIUM_CHANGED` ioctl is not supported.

The header file `<sys/scsi.h>` has useful information for direct access device control, including the following:

```
/* ioctl support for SCSI disk devices */
#define SIOC_GET_IR      _IOR('S', 14, int)
#define SIOC_SET_IR      _IOW('S', 15, int)
#define SIOC_SYNC_CACHE  -IOW('S', 70, int)
```

The `SIOC_FORMAT` ioctl reformats the entire media surface. Exclusive access to the device, obtained through use of the `DIOC_EXCLUSIVE` ioctl (see *disk(7)*), is required prior to reformatting to ensure that other applications are not affected. The `fmt_optn` field can be used to select the desired media geometry. Only one media geometry is supported on most devices. The value zero should be used for these devices. The value zero can also be used to select the default geometry on devices that support multiple media geometries. The `interleave` field can be used to specify sector interleaving. The value zero specifies that an appropriate default interleave should be used.

EXAMPLES

The following sample code shows how to use ioctls that affect `scsi_disk`.

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <sys/diskio.h>
#include <sys/scsi.h>
Describe (dfd)
int dfd;
```

```

{
    int ret;
    disk_describe_type descr_type;
    if ((ret = ioctl (dfd, DIOC_DESCRIBE, &descr_type)) != 0) {
        exit(1);
    }
    printf ("\nSuccessful ioctl DIOC_DESCRIBE \n");
    printf ("  model number: %s\n", descr_type.model_num);
    printf ("  interface:      %d <20=scsi>\n", descr_type.intf_type);
}
Exclusive (dfd)
    int dfd;
{
    int ret, flag=1;
    if ((ret = ioctl (dfd, DIOC_EXCLUSIVE, &flag)) != 0) {
        exit(1);
    }
}
Enable_WOE (dfd)
    int dfd;
{
    int ret, flag=1;
    if ((ret = ioctl (dfd, SIOC_WRITE_WOE, &flag)) != 0) {
        exit(1);
    }
    printf ("\nSuccessful ioctl SIOC_WRITE_WOE \n");
}
main (argc, argv)
    int argc;
    char ** argv;
{
    int ret, fd;
    if (argc != 2) {
        printf ("Usage: %s <disk3_device> \n", argv[0]);
        exit(1);
    }
    if ((fd = open (argv[1], O_RDWR)) < 0) {
        exit (1);
    }
    Describe (fd);
    Exclusive (fd);
    Enable_WOE (fd);
}

```

S

WARNINGS

Historically, disk devices have had small (typically 512 byte) block sizes; however, many newer disk devices (such as optical disks and disk arrays) have relatively large block sizes. Applications using direct raw disk access should use the `DIOC_DESCRIBE`, `DIOC_CAPACITY`, or `SIOC_CAPACITY` ioctl to determine the appropriate minimum I/O size.

Media removal and insertion while a disk device is open is unsupported and unpredictable. Do not attempt to circumvent prevention of media removal. Device capacity changes resulting from such intervention may not be recognized.

Often larger I/O operation sizes are expected to be more efficient. However, SCSI disk I/O operations that are large relative to the device's cache can result in insufficient cache space for the device to maintain full-media-speed data transfer rates. This can result in decreased I/O performance relative to smaller I/O sizes.

DEPENDENCIES**Optical Disk Devices**

The `SIOC_VERIFY_WRITES` ioctl controls the write mode. Normally written data is assumed to be correctly stored on the media. Verify-writes mode causes verification of written data to ensure that data has been correctly written. Verification can substantially reduce write performance and is not generally needed. The `SIOC_VERIFY_WRITES` ioctl can be used to enable or disable write verification. A zero value disables write verification. The value 1 enables write verification. Although write verification is

primarily intended for optical media, some systems may support write verification on normal disk devices.

The **SIOC_VERIFY** ioctl verifies that a media area contains valid data (that is, data that has been correctly written). Verified media will not cause I/O errors when reading is attempted. The media area to be verified is specified via the **start_lba** and **block_cnt** fields. Although verification is intended primarily for optical media, some systems may support verify operations on normal disk devices.

The **SIOC_WRITE_WOE** ioctl controls the write mode used for magneto-optical disk devices. Normally magneto-optical write operations require two physical head passes. The first pass erases the media area to be written. The second pass actually writes the data. Write-without-erase mode dramatically increases write performance by skipping the first (erase media area) pass. To ensure that the correct data results, it is essential that write-without-erase operations be performed only on media that is known to be blank (previously erased or never used). The **SIOC_WRITE_WOE** ioctl can be used to enable or disable write-without-erase. A zero value disables write-without-erase. The value 1 enables write-without-erase.

The **SIOC_ERASE** ioctl allows media areas to be explicitly erased. The media area to be erased is specified via the **start_lba** and **block_cnt** fields. Media areas erased in this manner can be written using write-without-erase mode. Note that an erased media area is different from a media area written with some data values (e.g. zeros). An erased media area should not be read. Attempting to read an erased media area generally results in an I/O error.

The **SIOC_VERIFY_BLANK** ioctl verifies that a media area has been erased and is suitable for being written using write-without-erase mode. The media area to be verified is specified via the **start_lba** and **block_cnt** fields.

The following optical disk device specific information is included from `<sys/scsi.h>`:

```
#define SIOC_WRITE_WOE      _IOW('S', 17, int)
#define SIOC_VERIFY_WRITES _IOW('S', 18, int)
#define SIOC_ERASE          _IOW('S', 19, struct scsi_erase)
#define SIOC_VERIFY_BLANK  _IOW('S', 20, struct scsi_verify)
#define SIOC_VERIFY        _IOW('S', 21, struct scsi_verify)

/* structure for SIOC_ERASE ioctl */
struct scsi_erase {
    unsigned int    start_lba;
    unsigned short  block_cnt;
};

/* structure for SIOC_VERIFY_BLANK and SIOC_VERIFY ioctls */
struct scsi_verify {
    unsigned int    start_lba;
    unsigned short  block_cnt;
};
```

FILES

`/usr/include/sys/scsi.h`

SEE ALSO

`mediainit(1)`, `mknod(1M)`, `ioctl(2)`, `disk(7)`, `scsi(7)`.

S

NAME

scsi_pt - SCSI pass-through device driver for Series 800 NIO based systems

DESCRIPTION

SCSI devices are normally controlled by a device-type-specific driver when the appropriate device-type-specific driver exists. Device-type-specific drivers, such as those for SCSI direct access (disk) and sequential access (tape) devices, coordinate device and driver states to accomplish correct logical device behavior. The SCSI pass-through driver (also called spt) enables use of SCSI devices and commands that are not supported by these device-type-specific drivers.

A successful `scsi_pt open()` call requires no device I/O operations. Once open, `ioctl()` calls can be used to change SCSI communication parameters, attempt SCSI commands, or other SCSI operations. Since the SCSI pass-through driver does not attempt to logically understand the target device, `read()` and `write()` calls are not supported. Super-user privileges or device write permissions are required to use these `ioctls`. All `reserved` fields in the data structures associated with these `ioctls` must be zero-filled.

Since the SCSI pass-through driver is a separate device driver, the default device-type-specific driver cannot be used for the same device at the same time as the pass-through driver.

SCSI Commands and Operations

Only two `ioctl` commands are supported by the NIO SCSI pass through driver; the `SIOC_EXCLUSIVE` command and the `SIOC_IO` command.

The `SIOC_EXCLUSIVE` `ioctl` provides a mechanism to prevent other opens of a device. This allows exclusive access by the locking application. There are two different levels of locks supported; SCSI target level and SCSI lun level. These levels are specified via the `data` field of the `ioctl()` call:

- 0 release exclusive access to SCSI lun
- 1 gain exclusive access to SCSI lun
- 2 release exclusive access to SCSI target
- 3 gain exclusive access to SCSI target

For example, to lock the device (SCSI target):

```
three = 3;
if (err = ioctl(fd, SIOC_EXCLUSIVE, &three)) {
    /* Could not lock the device -- check err */
}
```

NOTE: The corresponding unlock must be made for the acquired lock. An unlock of the SCSI target (`data == 2`) will NOT unlock all of the individual SCSI luns (`data == 0`).

The `SIOC_IO` `ioctl` allows an arbitrary SCSI command to be sent to a device. All details of the SCSI command protocol are handled automatically. See EXAMPLE mentioned below.

The header file `<sys/scsi.h>` has useful information for SCSI device control. The following is an excerpt from it:

```
/* SCSI device control ioctls */
#define SIOC_IO                _IOWR('S', 22, struct sctl_io)
#define SIOC_EXCLUSIVE         _IOW('S', 68, int)

/* Structure for SIOC_IO ioctl */
struct sctl_io
{
    unsigned        flags;
    unsigned char   cdb_length;
    unsigned char   cdb[16];
    void            *data;
    unsigned        data_length;
    unsigned        max_msecs;
    unsigned        data_xfer;
    unsigned        cdb_status;
    unsigned char   sense[256];
    unsigned        sense_status;
    unsigned char   sense_xfer;
    unsigned        reserved[16];
}
```

```
};
```

The **flags** field can have the following values (if supported):

SCTL_READ	Data-in phase expected if the data_length field is non-zero. The absence of this flag implies that a data-out phase is expected if the data_length field is non-zero.
SCTL_INIT_SDTR	synchronous data transfer request negotiations should be attempted with this command.
SCTL_INIT_WDTR	wide data transfer request negotiations should be attempted with this command.
SCTL_NO_ATN	device should be selected without attention (ATN). This implies that no SCSI message phase should be attempted with this command. (Not supported)
SCTL_2BYTE	device maintains 2-byte alignment. (Not supported)
SCTL_4BYTE	device maintains 4-byte alignment. (Not supported)

The **cdb_length** field specifies the number of valid command bytes in the **cdb** field.

The **cdb** field specifies the SCSI command bytes. These command bytes are sent to the target device during the SCSI command phase.

The **data** field specifies the address of the data area for the data phase of the SCSI command.

The **data_length** field specifies the maximum number of data bytes to be transferred. A zero value in the **data_length** field indicates that no data phase should occur. Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. The caller is responsible for correctly specifying both the **data_length** field and any **cdb** field data length values. Unexpected system behavior occurs if these fields are not consistent. The **data_length** value may not be larger than 1 Megabyte.

The **max_msecs** field specifies the maximum time, in milliseconds, that the device should need to complete the command. If this period of time expires without command completion, the system attempts recovery procedures to regain the device's attention. These recovery procedures may include device and bus reset operations. A zero value in the **max_msec** field indicates that the timeout period is infinite and the system should wait indefinitely for command completion. Note that very large (or infinite) timeout values can cause the SCSI bus (potentially the entire system) to "hang".

The **data_xfer** field indicates the number of data bytes actually transferred during the data phase of the cdb command. This field is valid only when the **cdb_status** field contains one of the following values: **S_GOOD**, **S_CHECK_CONDITION** or, **S_CONDITION_MET**.

The **cdb_status** field indicates the results of the cdb command. When the **SIOC_IO** ioctl call returns, all command processing has been completed. Most **SIOC_IO** ioctl calls will return zero (success). It is important, however, to always check the return value from the ioctl to ensure that an unexpected error has not occurred. The resulting detailed ioctl data should be used to evaluate "success" or "failure" from the caller's perspective. If the **cdb_status** field indicates a **S_CHECK_CONDITION** status, the **sense_status** field indicates the results of the SCSI **REQUEST SENSE** command used to collect the associated sense data. These status fields will contain one of the following values:

SCTL_INVALID_REQUEST	The SCSI command request is invalid and was not attempted.
SCTL_SELECT_TIMEOUT	The target device did not answer to selection by the host SCSI interface (the device does not exist or did not respond).
SCTL_INCOMPLETE	The device answered selection but the command was not completed (the device took too long or a communication failure occurred, ie. device power-fail).
SCTL_POWERFAIL	A system power recovery occurred. Error handling and retrying the command may be necessary. The pass-through driver does not perform any powerfail recovery.
S_GOOD	Device successfully completed the command.
S_CHECK_CONDITION	Device indicated sense data was available.

(Series 800 Only)

S_CONDITION_MET	Device successfully completed the command and the requested (search or pre-fetch) operation was satisfied.
S_BUSY	Device indicated it was unable to accept the command because it is busy doing other operations.
S_RESV_CONFLICT	Device indicated the command conflicted with an existing reservation.
S_COMMAND_TERMINATED	Device indicated the command was terminated early by the host system.
S_QUEUE_FULL	Device indicated it was unable to accept the command because its command queue is currently full.
S_INTERMEDIATE	Device successfully completed this command, which was one in a series of linked commands. (Not supported)
S_I_CONDITION_MET	Both S_INTERMEDIATE and S_CONDITION_MET indicated by the device. (Not supported)

The **sense** field indicates the sense data of the SCSI **REQUEST SENSE** command. It is only valid if the **sense_status** field indicates successful status.

The **sense_status** field indicates the results of the SCSI **REQUEST SENSE** command used to collect the associated sense data. It is only valid if the **cdb_status** field indicates a **S_CHECK_CONDITION** status.

The **sense_xfer** field indicates the number of valid sense data bytes. This field is valid only when the **cdb_status** field contains the value **S_CHECK_CONDITION** and the **sense_status** field contains the value **S_GOOD**.

EXAMPLE

The configuration test program provided with the SCSI pass-through driver product can be used as an example. It sends a SCSI **INQUIRY** command to the device referenced by the required special device file and prints out the returned inquiry data. The source for this program can be found in the file: `/usr/contrib/src/scsi_io.c`.

WARNINGS

- Incorrect use of SCSI device-control operations can easily cause data loss, resulting in a corrupted system.
- Only two **ioctl** commands are supported; the **SIOC_EXCLUSIVE** command and the **SIOC_IO** command. Any other commands attempted will return the **EINVAL** error.
- It is recommended that the **SIOC_EXCLUSIVE** **ioctl** be used to gain exclusive access to a device prior to attempting **SIOC_IO** commands. This will protect against other device opens from external sources (for example, from another user program). If exclusive access is not obtained, **SIOC_IO** commands may be intermixed between a number of different applications which can lead to undesirable results.
- Most SCSI commands have a logical unit number (LUN) field. SCSI implementations on the HP-UX operating system select logical units via the SCSI **IDENTIFY** message. The LUN portion of the **cdb** should normally be set to zero, even when the LUN being accessed is not zero.
- Most SCSI commands with a data phase expect the data length information to be included somewhere in the command bytes. Both the **data_length** field and any **cdb** data length values must be correctly specified to get correct command results.
- Very large timeout values can cause the SCSI bus and potentially the entire system to appear to be "hung".
- Unlike on the S700, the NIO SCSI pass-through driver is implemented as a stand-alone driver. To ensure system integrity, certain restrictions exist with its use of the bus and its interactions with other drivers.

Any device accessed via the NIO SCSI pass-through driver must be on a STAND-ALONE APPLICATION BUS. Therefore, no boot, dump, or swap devices can be connected to this bus. While a NIO SCSI pass-through driver application is running, it should have exclusive use of the application bus. There should be no other accesses to this bus, be it from the operating system, a user, or another application.

(Series 800 Only)

The controlling application can access its device(s) via the NIO SCSI pass-through driver and other HP standard driver(s) as long as the application handles the coordination between the various drivers. Thus, the application is responsible for initiating all communications to device(s) on this bus. For example, a tar command can not be directly used to a device on the exclusive application bus while the owning application is running. Any problems encountered must be displayed on this supported configuration (a stand-alone application bus) before support can be extended.

INSTALLATION AND CONFIGURATION

The SCSI pass-through file set comes with the core system and will exist after an install or an update. The files in this set (and their proper destinations) are:

Filename	Protections	Full path
=====	=====	=====
libspt.a	-r--r--r--	/usr/conf/lib/libspt.a
scsi_io.c	-r--r--r--	/usr/contrib/src/scsi_io.c
scsi_pt.7	-r--r--r--	/usr/share/man/man7.Z/scsi_pt.7
spt	-r--r--r--	/usr/conf/master.d/spt

The following configuration steps need to be completed to successfully use the NIO SCSI pass-through driver. This driver does not auto-configure.

Go to the build area:

```
$ cd /stand/build
```

Create a **system** file from your existing kernel:

```
$ /usr/sbin/sysadm/system_prep -s system
```

Get information about peripherals attached to your system:

```
$ ioscan -f
```

Check if the SCSI pass-through driver is already configured into your current kernel:

```
$ grep spt system
```

If no "spt" line was printed, add a line to the "SCSI drivers" area of the **system** file:

```
spt
```

A driver statement is needed to override the standard driver which gets autoconfigured for any device. One statement is required for each desired NIO SCSI pass-through device (be it a SCSI target, or a SCSI lun). Append the driver line(s) to the **system** file:

```
$ cat >> system
driver <path> spt
```

```
^D ... # control-D to exit the cat command
```

where *<path>* is the complete hardware path of desired device (from **ioscan** cmd)

Create the **conf.c** and **config.mk** files:

```
$ /usr/sbin/config -s system
```

Build a new kernel:

```
$ make -f config.mk
```

Verify that the pass-through driver has been built into the new kernel. (A line giving the revision information should be printed):

```
$ what ./vmunix_test | grep scsi_pt
```

Save the old **system** file:

```
$ mv /stand/system /stand/system.prev
```

Move the new **system** file to be the current one:

```
$ mv ./system /stand/system
```

S

Save the old kernel:

```
$ mv /stand/vmunix /stand/vmunix.prev
```

Move the new kernel to be the current one:

```
$ mv ./vmunix_test /stand/vmunix
```

Boot the system from the new kernel:

```
$ exec reboot
```

ONCE THE SYSTEM REBOOTS:

Verify the peripheral configuration:

```
$ ioscanner -f
```

Determine the major number for the SCSI pass-through driver:

```
$ lsdev -d spt
```

Create the special device file to access the scsi_pt peripheral (refer to the example in the TROUBLESHOOTING section):

```
$ mknod /dev/<devfilename> c <major #> <minor #>
```

where

<devfilename> name of the special device file

<major #> character major number (from lsdev cmd)

<minor #> minor number in the format 0xIITL00

where

II 2 digit card instance number (from **ioscanner** command; proper **ext_bus** entry)

T Target ID number (SCSI ID)

L Lun number (SCSI LUN)

00 Reserved fields, must be zero

All ready for use,try the sample program:

Compile the sample program:

```
$ cc /usr/contrib/src/scsi_io.c
```

Execute the sample program:

```
$ ./a.out /dev/<devfilename>
```

The sample program should have returned the inquiry data of the target device. If this did not happen, refer to the TROUBLESHOOTING section.

TROUBLESHOOTING

If the NIO SCSI pass-through driver is not working:

- Verify the proper configuration of the SCSI pass-through driver into the kernel.
- Verify that the hardware is set up properly.
 - Check that the device is connected, and that its SCSI Bus address is set properly.
 - Check the reference manual for the device to verify that it is configured properly.
 - Check that the SCSI Bus is properly terminated.
 - Verify that the device is operational using some other driver if possible. If a hardware problem is suspected, the SCSI pass-through driver must be unconfigured for that device and the HP specific device driver must be configured before any diagnostic software can be run.
- Verify the special device file.
 - Execute **lsdev -d spt** to find the major number for the SCSI pass-through driver. For example, **lsdev -d spt** might show:

(Series 800 Only)

- Character Block Driver Class
136 -1 spt spt
- Run **ioscan -kf** to find the card instance number, target SCSI ID and SCSI LUN for the desired device. For example **ioscan -kf** might show:

```

Class      I  H/W Path  Driver  S/W State H/W Type  Description
=====
bc          0              root    CLAIMED   BUS_NEXUS
ext_bus    2  8              scsi1    CLAIMED   INTERFACE HP 28655A - SC
target     0  8.3          target  CLAIMED   DEVICE
spt        0  8.3.0        spt     CLAIMED   DEVICE      HP      C1718T
target     1  8.4          target  CLAIMED   DEVICE
spt        1  8.4.0        spt     CLAIMED   DEVICE      HP      C1716T
...

```

In this example there are two desired devices:

```

Device #1                      Device #2
=====
card instance number = 2       card instance number = 2
target SCSI ID = 3             target SCSI ID = 4
SCSI LUN = 0                   SCSI LUN = 0

```

- Verify that these are the same values used in the special device files. For example, **ll /dev/c2t*d0** might show:

```

crw-rw-rw-  1 root  sys  136 0x023000 Dec 14 15:10 c2t3d0
crw-rw-rw-  1 root  sys  136 0x024000 Dec 14 15:10 c2t4d0

```

SEE ALSO

scsi(7), mknod(1M), uxgen(1M), reboot(2).

NAME

scsi_tape - SCSI sequential access (tape) device driver

DESCRIPTION

SCSI sequential-access (tape) devices store a sequence of data blocks. Data can be read and written using either fixed or variable sized block mode. If supported by the device, variable sized block mode is normally used (even when all blocks are the same size). Fixed sized block mode is generally only used for tape devices which do not support variable sized blocks. Fixed sized block mode can be used on some tape devices which support variable sized blocks to increase I/O performance.

Generally SCSI tape devices are controlled through the *mt(7)* generic tape device interface. This section describes features that are specific to SCSI tape devices.

The **SIOC_CAPACITY** ioctl (see *scsi(7)*) can be used to determine remaining tape capacity for some tape devices. The **blksz** field indicates the "natural" block size of the device. This value may or may not be the current block size of the device. The number of blocks, indicated by the **lba** field, is an estimate of how much data can be written on the remaining media. A zero size is returned for devices that do not provide remaining-capacity information. The quantity of data that can actually be written may be higher or lower than indicated, depending on such factors as block size, media defects, data compression, and ability to maintain streaming.

To improve performance, most SCSI tape devices have caches. Read-cache use, called "read ahead", causes the tape drive to read data in anticipation of read requests. Read ahead is only apparent to users in the increased performance that it produces. Write-cache use is called "immediate reporting". Immediate reporting increases I/O performance by reporting a completed write status before the data being written is actually committed to media. This allows the application program to supply additional data so that continuous media motion, called "streaming", can be achieved. The **SIOC_GET_IR** ioctl can be used to determine if immediate-reporting functionality is currently being used by the device. The value "1" indicates immediate reporting is enabled. By default, the device driver attempts to enable immediate reporting. The **SIOC_SET_IR** ioctl can be used to explicitly enable or disable immediate reporting. A zero value disables immediate reporting. The value "1" enables immediate reporting. The **MTIOCTOP** ioctl **MTNOP** command can be used to cause any cached data to be written (committed) to media. Note that the device immediate reporting mode set by the **SIOC_SET_IR** ioctl survives between **close()** and **open()** calls, but not through system reboot.

The **SIOC_GET_BLOCK_SIZE** ioctl indicates the device's current block size. A block size of zero indicates the device is in variable-sized-block mode. A non-zero block size indicates the device is in fixed-sized-block mode.

The **SIOC_SET_BLOCK_SIZE** ioctl changes the current block size to the specified number of bytes. Setting the block size to zero specifies that variable-sized-block mode should be used. Any non-zero block size specifies that fixed-sized-block mode should be used. By default, the device driver attempts to set the block size to zero during open. If variable-sized-block mode is not supported by the device, the driver selects an appropriate block size for fixed-sized-block mode use. Note that the device block size set by the **SIOC_SET_BLOCK_SIZE** ioctl survives between **close()** and **open()** calls, but not through system reboot.

The **SIOC_GET_BLOCK_LIMITS** ioctl indicates the device's maximum and minimum fixed block-size limits. The device's minimum fixed block size is indicated by the **min_blk_size** field. The **max_blk_size** field contains the smaller of the maximum block size supported by the device and the maximum block size supported by the system (**MAXPHYS**). This is the largest valid block size for the specific combination of device, driver, and host system being used.

The **SIOC_GET_POSITION** ioctl can be used to determine the current media position for some devices. For devices that support this capability, the resultant value can be used to reposition the media to the same position in the future.

The **SIOC_SET_POSITION** ioctl can be used to cause media repositioning on some devices. For devices that support this capability, media repositioning via this mechanism can generally be completed more quickly than might be similarly accomplished using record, filemark, or setmark spacing. The argument value specified should be the result of a previous **SIOC_GET_POSITION** for that media volume.

The following is included from **<sys/scsi.h>**:

```
/* ioctl support for SCSI tape commands */
#define SIOC_GET_IR          _IOR('S', 14, int)
#define SIOC_SET_IR          _IOW('S', 15, int)
```

(Series 700 Only)

```

#define SIOC_GET_BLOCK_SIZE      _IOR('S', 30, int)
#define SIOC_SET_BLOCK_SIZE      _IOW('S', 31, int)
#define SIOC_GET_BLOCK_LIMITS    _IOW('S', 32, struct scsi_block_limits)
#define SIOC_GET_POSITION        _IOR('S', 33, int)
#define SIOC_SET_POSITION        _IOW('S', 34, int)

/* structure for SIOC_GET_BLOCK_LIMITS ioctl */
struct scsi_block_limits {
    unsigned min_blk_size;
    unsigned max_blk_size;
};

```

WARNINGS

SCSI bus and device resets cause some devices to reposition media to beginning-of-tape (BOT). This unintentional media repositioning can cause loss of data. The `scsi_tape` driver causes the first subsequent `open()` attempt to fail as an indication of potential data loss.

The `scsi_tape` driver does not write filemarks at close if the media has been programatically repositioned. Applications that reposition the media prior to closing the device should write any required tape-marks.

SEE ALSO

`scsi(7)`, `mt(7)`, `mknod(1M)`.

NAME

socket - interprocess communications

DESCRIPTION

Sockets are communication endpoints that allow processes to communicate either locally or remotely. They are accessed by means of a set of system calls (see *socket(2)*).

The following *ioctl*() requests are defined in *<sys/ioctl.h>* (see *ioctl(2)*):

- FIONBIO** If the int with the address *arg* is non-zero, the socket is put into non-blocking mode. Otherwise, the socket is put into blocking mode. Blocking mode is the default. The **FIONBIO** request is equivalent to the **FIOFNIO** request, although using **FIONBIO** is not recommended. See *accept(2)*, *connect(2)*, *recv(2)*, and *send(2)* for an explanation of how non-blocking mode is used.
- FIONREAD** For SOCK_STREAM sockets, the number of bytes currently readable from this socket is returned in the integer with the address *arg*. For SOCK_DGRAM sockets, the number of bytes currently readable, plus the size of the *sockaddr* structure (defined in *<sys/socket.h>*), is returned in the integer with the address *arg*.
- SIOCATMARK** For SOCK_STREAM TCP sockets, on return the integer with the address *arg* is non-zero if the inbound TCP stream has been read up to where the out-of-band data byte starts. Otherwise, the inbound TCP stream has not yet been read up to where the out-of-band data byte starts. For sockets other than SOCK_STREAM TCP sockets, on return the integer with the address *arg* is always zero.
- SIOCSPGRP** This request sets the process group or process ID associated with the socket to be the value of the integer with the address *arg*. A process group or process ID associated with the socket in this manner is signaled when the state of the socket changes: **SIGURG** is delivered upon the receipt of out-of-band data; **SIGIO** is delivered if the socket is asynchronous, as described in **FIOASYNC** below. If the value of the integer with the address *arg* is positive, the signal is sent to the process whose process ID matches the value specified. If the value is negative, the signal is sent to all the processes that have a process group equal to the absolute value of the value specified. If the value is zero, no signal is sent to any process. It is necessary to issue this request with a non-zero integer value to enable the signal delivery mechanism described above. The default for the process group or process ID value is zero.
- SIOCGPGRP** This request returns the process group or process ID associated with the socket in the integer with the address *arg*. See the explanation for **SIOCSPGRP** above for more details on the meaning of the integer value returned.
- FIOASYNC** If the integer whose address is *arg* is non-zero, this request sets the state of the socket as asynchronous. Otherwise, the socket is put into synchronous mode (the default). Asynchronous mode enables the delivery of the **SIGIO** signal when:

- New data arrives, or
- For connection-oriented protocols, whenever additional outgoing buffer space becomes available or when the connection is established or broken.

The process group or process ID associated with the socket must be non-zero in order for **SIGIO** signals to be sent. The signal is delivered according to the semantics of **SIOCSPGRP** described above.

The *fcntl(2)* **O_NDELAY** and **O_NONBLOCK** flags (defined in *<fcntl.h>*) are supported by sockets. If the **O_NONBLOCK** flag is set, the socket is put into POSIX-style non-blocking mode. If the **O_NDELAY** flag is set, the socket is put into non-blocking mode. Otherwise, the socket is put into blocking mode. Blocking mode is the default. See *accept(2)*, *connect(2)*, *recv(2)*, and *send(2)* for an explanation of how these forms of non-blocking mode are used.

Since the *fcntl*() **O_NONBLOCK** and **O_NDELAY** flags and *ioctl*() **FIONBIO** requests are supported, the following clarifies on how these features interact. If the **O_NONBLOCK** or **O_NDELAY** flag has been set, *recv*() and *send*() requests behave accordingly, regardless of any **FIONBIO** requests. If neither the **O_NONBLOCK** flag nor the **O_NDELAY** flag has been set, **FIONBIO** requests control the behavior of *recv*() and *send*().

DEPENDENCIES**AF_CCITT Only**

Only the **FIOSNBIO**, **FIONREAD**, **SIOCGPGRP**, and **SIOCSPGRP** `ioctl()` requests are defined for **af_ccitt** sockets. See *socketx25(7)*.

AUTHOR

socket was developed by the University of California, Berkeley.

SEE ALSO

`fcntl(2)`, `getsockopt(2)`, `ioctl(2)`, `socket(2)`, `socketx25(7)`.

NAME

streamio - STREAMS ioctl commands

SYNOPSIS

```
#include <sys/types.h>
#include <stropts.h>

int ioctl(int fildes, int command, ... /* arg */);
```

DESCRIPTION

STREAMS **ioctl** commands are a subset of the **ioctl()** system calls which perform a variety of control functions on streams.

fildes is an open file descriptor that refers to a stream. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure. The *command* and *arg* are interpreted by the stream head. Certain combinations of these arguments may be passed to a module or driver in the stream.

Since these STREAMS commands are a subset of **ioctl**, they are subject to the errors described there. In addition to those errors, the call will fail with **errno** set to **EINVAL**, without processing a control function, if the stream referenced by *fildes* is linked below a multiplexor, or if *command* is not a valid value for a stream.

Also, as described in **ioctl**, STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the stream head containing an error value. This causes subsequent system calls to fail with **errno** set to this value.

The following **ioctl** commands, with error values indicated, are applicable to all STREAMS files:

I_ATMARK Allows the user to see if the current message on the stream head read queue is "marked" by some module downstream. *arg* determines how the checking is done when there are multiple marked messages on the stream head read queue. It may take the following values:

ANYMARK Checks if the message is marked.

LASTMARK Checks if the message is the last one that is marked on the queue.

If both **ANYMARK** and **LASTMARK** are set, **ANYMARK** supersedes **LASTMARK**.

The return value is 1 if the mark condition is satisfied and 0 otherwise.

I_CANPUT Checks if a certain band is writable. *arg* is set to the priority band in question. The return value is 0 if the priority band *arg* is flow controlled, 1 if the band is writable, or -1 on error.

I_CKBAND Check if the message of a given priority band exists on the stream head read queue. This returns 1 if a message of a given priority exists, or -1 on error. *arg* should be an integer containing the value of the priority band in question.

I_FDINSERT Creates a message from user specified buffer(s), adds information about another stream and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a **strfdinsert** structure which contains the following members:

```
struct strbuf    ctlbuf;
struct strbuf    databuf;
long             options;
int              fildes;
int              offset;
```

The *len* field in the **ctlbuf strbuf** structure (see *putmsg(2)*) must be set to the size of a pointer plus the number of bytes of control information to be sent with the message. *fildes* in the **strfdinsert** structure specifies the file descriptor of the other stream. *offset*, which must be word-aligned, specifies the number of bytes beyond the beginning of the control buffer where **I_FDINSERT** will store a pointer. This pointer will be the address of the read queue structure of the driver for the

streams corresponding to *filde*s in the **strfdinsert** structure. The *len* field in the **datbuf** **strbuf** structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

flags specifies the type of message to be created. An ordinary (non-priority) message is created if *flags* is set to 0, a high priority message is created if *flags* is set to **RS_HIPRI**. For normal messages, **I_FDINSERT** will block if the stream write queue is full due to internal flow control conditions. For high priority messages, **I_FDINSERT** does not block on this condition. For normal messages, **I_FDINSERT** does not block when the write queue is full and the **O_NONBLOCK** is set. Instead, it fails and sets **errno** to **EAGAIN**.

I_FDINSERT also blocks, unless prevented by the lack of internal resources, waiting for the availability of message blocks, regardless of priority or whether **O_NONBLOCK** has been specified. No partial message is sent.

I_FDINSERT can also fail if an error message was received by the stream head of the stream corresponding to *filde*s in the **strfdinsert** structure. In this case, **errno** will be set to the value in the message.

I_FIND Compares the names of all modules currently present on the stream to the name specified in *arg*. The command returns a value of 1 if the module is present and a value of 0 (zero) if the module is not present.

I_FLUSH This request flushes all input and/or output queues, depending on the value of *arg*. Valid *arg* values are:

FLUSHRW	Flush write and read queues.
FLUSHW	Flush write queues.
FLUSHR	Flush read queues.

If a pipe or FIFO does not have any modules pushed, the read queue of the streams head on either end is flushed depending on the value of *arg*.

If **FLUSHR** is set and *filde*s is a pipe, the read queue for that end of the pipe is flushed and the write queue for the other end is flushed. If *filde*s is a FIFO, both queues are flushed.

If **FLUSHW** is set and *filde*s is a pipe and the other end of the pipe exists, the read queue for the other end of the pipe is flushed and the write queue for this end is flushed. If *filde*s is a FIFO, both queues of the FIFO are flushed.

If **FLUSHRW** is set, all read queues are flushed, that is the read queue for the FIFO and the read queue of both ends of the pipe are flushed.

Correct flushing handling of a pipe or FIFO with modules pushed is achieved via the **pipemod** module. This module should be the first module pushed onto a pipe so that it is at the midpoint of the pipe itself.

I_FLUSHBAND

Flushes a particular band of messages. *arg* points to a **bandinfo** structure that has the following members:

unsigned char	bi_pri:
int	bi_option;

The value of the *bi_option* field can be **FLUSHR**, **FLUSHW**, or **FLUSHRW** as described for the **I_FLUSH** command.

I_GETBAND Returns the priority band of the first message on the stream head read queue in the integer referenced by *arg*.

I_GETCLTIME

Returns the close time delay in the long pointed by *arg*.

I_SETCLTIME

Allows the user to set the time that the stream head will delay when a stream is closing, and there is data on the write queues. Before closing each module and driver, the stream head will delay for the specified amount of time to allow the data to drain. If, after the delay, data is still present, data will be flushed. *arg* is a pointer to the

number of milliseconds to delay, rounded up to the nearest valid value on the system. The default is fifteen seconds.

- I_GETSIG** Returns the events for which the calling process has registered to receive a **SIG-POLL** signal. Events are returned as in *arg* bitmask as defined for the **I_SETSIG** command.
- I_GRDOPT** Returns the current read mode setting in an *int* pointed to by the argument *arg*. Read modes are described in *read(2)*.
- I_GWROPT** Returns the current write mode setting, as described in **I_SWROPT**, in the *int* that is pointed to by the argument *arg*.
- I_LINK** Connects two streams, where *fildev* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* gets connected below the multiplexing driver. **I_LINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call returns a multiplexor ID number (an identifier used to disconnect the multiplexor, see **I_UNLINK**) on success, and -1 on failure.
- I_LIST** Allows the user to list all the module names on the stream, up to and including the topmost driver name. If *arg* is **NULL**, the return value is the number of modules, including the driver, that are on the stream pointed to by *fildev*. This allows the user to allocate enough space for the module names. If *arg* is not **NULL**, it should point to a **str_list** structure that has the following members:

```
int      sl_nmods;
struct   str_mlist  *sl_modlist;
```

The **str_mlist** structure has the following member:

```
char     l_name[FMNAMESZ+1];
```

sl_nmods indicates the number of entries the user has allocated in the array. On success, the return value is 0, **sl_modlist** contains the list of module names, and **sl_nmods** indicates the number of entries that have been filled in.

- I_LOOK** Retrieves the name of the module located just below the stream's head of the stream pointed to by *fildev*, and places it in a null terminated character string pointed at by *arg*. The buffer pointed to by *arg* should be at least **FMNAMESZ+1** bytes long. A **#include <stropts.h>** declaration is required.
- I_NREAD** Counts the number of data bytes in data blocks in the first message on the stream head read queue, and places this value in the location pointed to by *arg*. The return value for the command is the number of messages on the stream head read queue. For example, if zero is returned in *arg*, but the **ioctl** return value is greater than zero, this indicates that a zero-length message is next on the queue.
- I_PEEK** Allows the user process to look (peek) at the contents of the first message on the stream head read queue. This is done without taking the message off the queue. The **I_PEEK ioctl** operates the same way as the **getmsg()** function, except that it does not remove the message. The *arg* parameter points to a **strpeek** structure (in the **<stropts.h>** header file) with the following members:

```
struct strbuf  ctlbuf;
struct strbuf  databuf;
long           options;
```

The **strbuf** structure pointed to by **ctlbuf** and **databuf** has the following members:

```
int    maxlen;
int    len;
char   *buf
```

The *maxlen* field of the **strbuf** structure must specify the number of bytes of control or data information to be retrieved. The *options* field can be set to **RS_HIPRI** or 0 (zero). If this field is set to **RS_HIPRI**, the **I_PEEK ioctl** looks for a high priority message on the queue. If the field is set to 0, the **I_PEEK ioctl** looks at

the first message on the queue.

The **I_PEEK** returns a 1 if a message was retrieved, and returns a value of 0 (zero) if no message was found; it does not wait for a message. Upon successful completion, **ctlbuf** specifies control information in the control buffer, **databuf** specifies data information in the data buffer, and **options** contains **RS_HIPRI** or 0 (zero).

I_PLINK Connects two streams, where *fildev* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* gets connected via a persistent link below the multiplexing driver. **I_PLINK** requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call creates a persistent link which can exist even if the file descriptor associated with the upper stream to the multiplexing driver is closed. This call returns a multiplexor ID number (an identifier that may be used to disconnect the multiplexor, see **I_PUNLINK**) on success and -1 on failure.

The **I_PLINK** **ioctl** can also fail if it is waiting for the multiplexing driver to acknowledge the link request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) message is received at the stream head for *fildev*. In addition, an error can be returned in an **M_IOACK** or **M_IONAK** message. When these occur, the **I_PLINK** fails with **errno** set to the value in the message.

I_POP Removes the module just below the stream head of the stream pointed to by *fildev*. To remove a module from a pipe requires that the module was pushed on the side it is being removed from. *arg* should be 0 in an **I_POP** request.

I_PUNLINK Disconnects the two streams specified by *fildev* and *arg* that are connected with a persistent link. *fildev* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by **I_PLINK** when a stream was linked below the multiplexing driver. If *arg* is **MUXID_ALL**, then all streams which are persistent links to *fildev* are disconnected. As in **I_PLINK**, this command requires the multiplexing driver to acknowledge the unlink.

I_PUSH Pushes the module whose name is pointed by *arg* onto the top of the current stream, just below the stream head. If the stream is a pipe, the module will be inserted between the streams heads of both ends of the pipe. It then calls the open routine of the newly-pushed module.

I_RECVFD Retrieves the file descriptor associated with the message sent by an **I_SENDFD** **ioctl** over a stream pipe. *arg* is a pointer to a data buffer large enough to hold a **strrecvfd** data structure containing the following members:

```
int      fd;
uid_t    uid;
gid_t    gid;
char     fill[8]
```

fd is an integer file descriptor. **uid** and **gid** are the user ID and group ID, respectively, of the sending stream.

If **O_NONBLOCK** is clear, **I_RECVFD** will block until a message is present at the stream head. If **O_NONBLOCK** is set, **I_RECVFD** will fail with **errno** set to **EAGAIN** if no message is present at the stream head.

If the message at the stream head is a message sent by a **I_SENDFD**, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the *fd* field of the **strrecvfd** structure. The structure is copied into the user data buffer pointed to by *arg*.

I_SENDFD Requests the stream associated with *fildev* to send a message, containing a file pointer, to the stream head at the other end of a stream pipe. The file pointer corresponds to *arg*, which must be an open file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user ID and group ID associated with the sending process are also inserted. This message is placed directly on the read queue of the stream head at the other end of the stream pipe to which it is connected.

S

I_SETCLTIME

Lets the user process set the time that the stream head delays when the stream is closing and the write queues contain data. The *arg* parameter contains a pointer to the number of milliseconds to delay, rounded up to the nearest legal value on the system. The default time is 15 seconds.

Before STREAMS modules and drivers are closed, the stream head delays for the specified amount of time. This allows the data on the write queues to drain. If data is still present on the writes queues after the delay, the queues are flushed.

I_SETSIG

Informs the stream head that the user wants the kernel to issue the **SIGPOLL** signal (see *signal(2)*) when a particular event has occurred on the stream associated with *files*. **I_SETSIG** supports an asynchronous processing capability in STREAMS. The value of *arg* is a bitmask that specifies the events for which the user should be signaled. It is the bitwise-OR of any combination, except where noted, of the following constants:

S_BANDURG	When used in conjunction with S_RDBAND , SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the stream head read queue.
S_ERROR	An M_ERROR message has reached the stream head.
S_HANGUP	An M_HANGUP message has reached the stream head.
S_HIPRI	A high priority message is present on the stream head read queue. This is set even if the message is of zero length.
S_INPUT	Any message other than an M_PCPROTO has arrived on a stream head read queue. This event is maintained for compatibility with prior releases. This is set even if the message is of zero length.
S_MSG	A STREAMS signal message that contains the SIGPOLL signal has reached the front of the stream head read queue.
S_OUTPUT	The write queue just below the stream head is no longer full. This notifies the user that there is room on the queue for sending (or writing) data downstream.
S_RDBAND	A priority band message (band > 0) has arrived on a stream head read queue. This is set even if the message is of zero-length.
S_RDNORM	An ordinary (non-priority) message has arrived on a stream head read queue. This is set even if the message is of zero-length.
S_WRBAND	A priority band greater than 0 of a queue downstream exists and is writable. This notifies the user that there is room on the queue for sending (or writing) priority data downstream.
S_WRNORM	This event is the same as S_OUTPUT .

A user process may choose to be signaled only of high priority messages by setting *arg* bitmask to the value **S_HIPRI**.

Processes that want to receive **SIGPOLL** signals must explicitly register to receive them using **I_SETSIG**. If several processes register to receive the signal for the same event on the same stream, each process will be signaled when the event occurs.

If the value of *arg* is zero, the calling process will be unregistered and will not receive further **SIGPOLL** signals.

I_SRDOPT

Sets the read mode (see *read(2)*) using the value of the argument *arg*. Valid *arg* values are:

RNORM	Byte-stream mode (default).
RMSGD	Message-discard mode.
RMSGN	Message-nondiscard mode.

Setting both **RMSGD** and **RMSGN** is an error. **RMSGD** and **RMSGN** override **NORM**.

In addition, treatment of control messages by the stream head may be changed by setting the following flags in *arg*:

- RPROTNORM** Fail **read** with EBADMSG if a control message is at the front of the stream head read queue. This is the default behavior.
- RPROTDAT** Deliver the control portion of a message as data when a user issues **read**.
- RPROTDIS** Discard the control portion of a message, delivering any data portion, when a user issues a **read**.

I_STR Constructs an internal STREAMS **ioctl** message from the data pointed to by *arg*, and sends that message downstream.

This mechanism is provided to send user **ioctl** requests to downstream modules and drivers. It allows information to be sent with the **ioctl**, and will return to the user any information sent upstream by the downstream recipient. **I_STR** blocks until the system responds with either a positive or negative acknowledgement message, or until the request "times out" after some period of time. If the request times out, it fails with **errno** set to ETIME.

At most, one **I_STR** can be active on a stream. Further **I_STR** calls will block until the active **I_STR** completes at the stream head. The default timeout intervals for these requests is 15 seconds. The **O_NONBLOCK** (see *open(2)*) flags have no effect on this call.

To send requests downstream, *arg* must point to a **strioc** structure which contains the following members:

```
int      ic_cmd;
int      ic_timeout;
int      ic_len;
char     *ic_dp;
```

ic_cmd is the internal **ioctl** command intended for the downstream module or driver and **ic_timeout** is the number of seconds (-1 =infinite, 0 = use default, >0 = as specified) an **I_STR** request will wait for acknowledgement before timing out. The default timeout is infinite. **ic_len** is the number of bytes in the data argument and **ic_dp** is a pointer to the data argument. The **ic_len** field has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the user (the buffer pointed to by **ic_dp** should be large enough to contain the maximum amount of data that any module or driver in the stream can return). The stream head will convert the information pointed to by **strioc** structure to an internal **ioctl** command message and send it downstream.

I_SWROPT Sets the write mode using the value of the argument *arg*. Legal bit settings for *arg* are:

- SNDZERO** Sends a zero-length message downstream when a write of 0 bytes occurs. To not send a zero-length message when a write of 0 bytes occurs, this bit must not be set in *arg*.

I_UNLINK Disconnects the two streams specified by *fil* and *arg*. *fil* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexor ID number that was returned by the **I_LINK**. If *arg* is **MUXID_ALL**, then all streams which were linked to *fil* are disconnected. As in **I_LINK**, this command requires the multiplexing driver to acknowledge the unlink.

RETURN VALUE

Unless specified differently for a command, the return value for a STREAMS **ioctl**() call is 0 (zero) on success and -1 (minus one) on failure.

S

ERRORS

A STREAMS **ioctl** command fails without performing the function and with **errno** set to **[EINVAL]** if:

- The stream referred to by *fildev* is linked below a multiplexing driver.
- The *command* parameter is not a valid value for the stream.

In addition, if any of the following conditions occur, the STREAMS **ioctl** commands return the corresponding value:

I_ATMARK

[EINVAL] *arg* has an illegal value.

I_CANPUT

[EINVAL] *arg* has an illegal value.

I_CKBAND

[EINVAL] *arg* has an illegal value.

I_FDINSERT

[EINVAL] The *fildev* parameter in the **strfdinsert** structure is an invalid open file descriptor.

[EINVAL] The size of the pointer plus *offset* exceeds the value of the *len* field for the buffer specified through *ctlptr*.

[EINVAL] *offset* does not specify a properly aligned location in the data buffer.

[EINVAL] *options* contains an undefined value.

[EFAULT] *arg* points, or **ctrlbuf** or **databuf** is outside the allocated address space.

[EAGAIN] The **ioctl** request failed because a non-priority message was to be created, the **O_NONBLOCK** option was set, and the stream's write queue was full because of internal flow control conditions.

[ENOSR] Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.

[ENXIO] A hangup was received on the stream specified by *fildev* in the **I_FDINSERT** **ioctl** call or on the stream specified by *fildev* in the **strfdinsert**.

[ERANGE] The value of the *len* field for the buffer specified through **databuf** does not fall within the range for the minimum and maximum sizes of packets for the top-most module on the stream.

[ERANGE] The value of the *len* field for the buffer specified through **databuf** is larger than the maximum allowable size for the data part of a message.

[ERANGE] The value of the *len* field for the buffer specified through **ctrlbuf** is larger than the maximum allowable size for the control part of a message.

The **I_FDINSERT** **ioctl** can also fail if an error (**M_ERROR**) message was received by the stream specified by the *fildev* field in the **strfdinsert** structure. In this case, **errno** is set to the error value in the error message.

I_FIND

[EINVAL] *arg* does not contain a valid module name.

[EFAULT] *arg* points outside the allocated address space.

I_FLUSH

[ENOSR] Could not allocate buffers for flush operation because of a lack of STREAMS memory resources.

[EINVAL] The *arg* parameter is an invalid value.

[ENXIO] A hangup was received on *fildev*.

I_FLUSHBAND

[EINVAL]	The <i>bi_pr</i> parameter value exceeds the maximum band, or the <i>bi_option</i> parameter is not FLUSHR , FLUSHW , or FLUSHRW .
I_GETBAND	
[ENODATA]	No message exists on the stream head read queue.
I_GETSIG	
[EINVAL]	User process is not registered to receive the SIGPOLL signal.
[EFAULT]	<i>arg</i> points outside the allocated address space.
I_GRDOPT	
[EFAULT]	<i>arg</i> is pointing outside the allocated address space.
I_LINK	
[EAGAIN]	Temporarily unable to allocate storage to perform the linking operation.
[EBADF]	The <i>arg</i> parameter not a valid open file descriptor.
[ENXIO]	A hangup was received on <i>fildev</i> .
[EINVAL]	The stream referred to by <i>fildev</i> does not support multiplexing.
[EINVAL]	The file referred to by <i>arg</i> is not a stream, or the stream is already linked under a multiplexor.
[EINVAL]	The link operation would cause a "cycle" in the resulting multiplexing configuration. In other words, the driver referred to by the <i>arg</i> parameter is linked into this configuration at multiple places
[ENOSR]	Not enough STREAMS memory resources to allocate storage for this command.
[ETIME]	Acknowledgement message not received at stream head before timeout.
The I_LINK ioctl can also fail if an M_ERROR or M_HANGUP message is received at the stream head for <i>fildev</i> before receiving the driver acknowledgement. In addition, an error can be returned in an M_IOCACK or M_IOCNAK message. When these occur, the I_LINK ioctl fails with errno set to the value in the message.	
I_LIST	
[EINVAL]	sl_nmods is less than 1.
[EAGAIN]	Could not allocate buffers.
I_LOOK	
[EINVAL]	There are no modules in the stream.
[EFAULT]	<i>arg</i> points outside the allocated address space.
I_NREAD	
[EFAULT]	<i>arg</i> is pointing outside the allocated address space.
I_PEEK	
[EINVAL]	The <i>options</i> parameter is an illegal value.
[EFAULT]	<i>arg</i> points, or ctrlbuf or databuf is, outside the allocated address space.
[EBADMSG]	Message to be looked at is not valid for the I_PEEK command.
I_PLINK	
[ENXIO]	A hangup was received on the stream referred to by the <i>fildev</i> parameter.
[ETIME]	A timeout occurred before an acknowledgement message was received at the stream head.
[EAGAIN]	Temporarily unable to allocate storage to perform the linking operation.
[EBADF]	<i>arg</i> is not a valid open file descriptor.
[EINVAL]	The stream referred to by <i>fildev</i> does not support multiplexing.

[EINVAL] The file referred to by *arg* is not a stream or is already linked under a multiplexing driver.

[EINVAL] The link operation would cause a "cycle" in the resulting multiplexing configuration. In other words, the driver referred to by *arg* is linked into the configuration at multiple places.

I_POP

[EINVAL] There are not modules in the stream.

[ENXIO] Error value returned by the module being popped.

[ENXIO] A hangup was received on *fildev*.

I_PUNLINK

[ENXIO] A hangup was received on *fildev*.

[ETIME] A timeout occurred before an acknowledgement message was received at the stream head.

[EAGAIN] Temporarily unable to allocate storage to perform the linking operation.

[EINVAL] *arg* is an invalid multiplexor ID number.

[EINVAL] *fildev* is the file descriptor of a pipe.

An **I_PUNLINK** *ioctl* can also fail if it is waiting for the multiplexor to acknowledge the unlink request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) is received at the stream head for *fildev*. In addition, an error can be returned in an **M_IOCACK** or **M_IOCNAK** message. When these occur, the **P_UNLINK** *ioctl* fails with **errno** set to the value in the message.

I_PUSH

[EINVAL] An invalid module name was used.

[EFAULT] *arg* points outside the allocated address space.

[ENXIO] Error value returned by the module being pushed. The push has failed.

[ENXIO] A hangup was received on *fildev*.

I_RECVFD

[EAGAIN] The **O_NONBLOCK** option was set, and a message was not present on the stream head read queue.

[EFAULT] The *arg* parameter points outside the allocated address space.

[EBADMSG] The message present on the stream head read queue did not contain a passed file descriptor.

[EMFILE] Too many open files. No more file descriptors are permitted to be opened.

[ENXIO] A hangup was received on *fildev*.

I_SENDFD

[EAGAIN] The sending stream head could not allocate a message block for the file pointer.

[EAGAIN] The read queue of the receiving stream head was full and could not accept the message.

[EBADF] The *arg* parameter is not a valid open file descriptor.

[EINVAL] The *fildev* parameter does not refer to a stream.

[ENXIO] A hangup was received on *fildev*.

I_SETCLTIME

[EINVAL] *arg* has an illegal value.

I_SETSIG

[EINVAL] The user process is not registered to receive the **SIGPOLL** signal.

[EAGAIN]	A data structure to store the signal request could not be allocated.
I_SRDOPT	
[EINVAL]	<i>arg</i> contains an illegal value.
I_STR	
[EINVAL]	The ic_len field is less than 0 (zero) bytes or larger than the maximum allowable size of the data part of a message (ic_dp).
[EINVAL]	The ic_timeout field is less than -1.
[EFAULT]	<i>arg</i> points, or the buffer area specified by ic_dp or ic_len is, outside the allocated address space.
[ENOSR]	Buffers could not be allocated for the ioctl request because of a lack of STREAMS memory resources.
[ENXIO]	A hangup was received on the stream referred to by <i>fildev</i> .
[ETIME]	The ioctl request timed out before an acknowledgement was received.

The **I_STR ioctl** can also fail if the stream head receives a message indicating an error (**M_ERROR**) or a hangup (**M_HANGUP**). In addition, an error can be returned in an **M_IOCACK** or **M_IOCNAK** message. In these cases, the **ioctl** fails with **errno** set to the error value in the message.

I_SWROPT

[EINVAL]	The <i>arg</i> parameter is an illegal value.
----------	---

I_UNLINK

[ENXIO]	A hangup was received on <i>fildev</i> .
[ETIME]	A timeout occurred before an acknowledgement message was received at the stream head.
[EINVAL]	<i>arg</i> is an invalid multiplexor ID number, or <i>fildev</i> is already linked under a multiplexing driver.

An **I_UNLINK ioctl** can also fail if it is waiting for the multiplexor to acknowledge the unlink request and an error (**M_ERROR**) message, or hangup (**M_HANGUP**) is received at the stream head for *fildev*. In addition, an error can be returned in **M_IOCACK** or **M_IOCNAK** message. When this occurs, the **I_UNLINK ioctl** fails with **errno** set to the value in the message.

SEE ALSO

close(2), fcntl(2), getmsg(2), ioctl(2), open(2), poll(2), putmsg(2), read(2), write(2), signal(5).

NAME

strlog - STREAMS log driver

DESCRIPTION

The STREAMS log driver allows user-level processes and STREAMS drivers and modules to perform error logging and event tracing. These tasks are done via a user interface and a kernel interface. Further, the STREAMS log driver delivers error logging and event tracing messages to the Network Tracing and Logging Facility (NetTL) (see *nettl(1M)*, *netfmt(1M)*, and *nettlconf(1M)*).

The interface that this driver presents to user-level processes is a subset of the *ioctl()* system calls and STREAMS message formats. These processes can be error loggers, trace loggers, or other user processes, that generate error or event messages. The user interface collects log messages from the log driver, and also generates log messages from user processes.

The driver also accepts log messages from STREAMS drivers and modules in the kernel via its function call interface. The kernel interface enters requests or calls from STREAMS drivers and modules into log messages.

The log messages accepted by the log driver are also delivered to NetTL. NetTL can be used to control which types of messages to log, and to format and filter the logged messages.

Kernel Interface

STREAMS drivers and modules generate log messages by calls to the **strlog** function.

```
#include <sys/strlog.h>
```

```
int strlog (mid, sid, level, flags, fmt [, value ]...);
short mid;
short sid;
char level;
ushort flags;
char *fmt;
int value;
```

mid specifies the STREAMS module ID number for the driver or module submitting the log message.

sid specifies the sub-ID number of a minor device associated with the STREAMS module or driver identified by *mid*.

level specifies a level for screening lower-level event messages from a tracer.

flags contains several flags that can be set in various combinations. The flags are as follows:

SL_ERROR The message is for the error logger.

SL_TRACE The message is for the tracer.

SL_CONSOLE The message will be printed to the console.

SL_FATAL Provides a notification of a fatal error.

SL_NOTIFY Makes a request to mail a copy of a message to the system administrator.

The following are additional flags. These flags are not used by **strerr** or **strace**. However, they are used to map STREAMS messages to NetTL messages as described below in "STREAMS-NetTL Link" section.

SL_WARN The message is a warning.

SL_NOTE The message is a note.

fmt is a **printf** style format string. This accepts the **%x**, **%l**, **%o**, **%u**, **%d**, **%c**, and **%s** conversion specifications.

values are numeric or character arguments for the format string. There is no maximum number of arguments that can be specified.

User Interface

User processes access the log driver with an **open()** call to **/dev/strlog**. Each open to the device will obtain a separate stream. After a process opens **/dev/strlog**, it indicates whether it is an error logger

or trace logger. It does this by issuing an `I_STR ioctl()` system call with the appropriate value in the `ic_cmd` field of the `striocctl` structure, and the appropriate data and control information in a `trace_ids` structure:

```
struct trace_ids {
    short    ti_mid;
    short    ti_sid;
    char     ti_level;
    short    ti_flags;
};
```

The values for `ic_cmd` are:

- I_ERRLOG** Indicates an error logger. No `trace_ids` data is needed.
- I_TRCLOG** Indicates a trace logger. A data buffer consisting of an array of one or more `trace_ids` structures must be included.

If any of the fields of the `trace_ids` structure contain a value of -1, `/dev/strlog` will accept whatever value it receives in that field. Otherwise, `strlog` only accepts messages only if the values of `mid` and `sid` are the same as their counterparts in the `trace_ids` structure, and if the message's level is equal to or less than the `level` value in the `trace_ids` structure.

Once the logger process has sent the `I_STR ioctl()` call, the STREAMS log driver begins to send log messages matching the restrictions to the logger process. The logger process obtains the log messages via the `getmsg()` system call. The control part of the messages passed in this call includes a `log_ctl` structure:

```
struct log_ctl {
    short    mid;
    short    sid;
    char     level;
    short    flags;
    long     ltime;
    long     ttime;
    int      seq_no;
};
```

The `log_ctl` structure indicates the `mid`, `sid`, and `level` time in ticks since the boot time that the message was submitted, the corresponding time in seconds since January 1, 1970, and a sequence number. The time in seconds since January 1, 1970 is provided so that the date and time of the message can be easily computed. The time in ticks since boot time is provided so that the relative timing of log messages can be determined.

A user process, other than an error or trace logger, can send a log message to `strlog`. The driver will accept only the `flags` and `level` fields of the `log_ctl` structure in the control part of the message, and a properly formatted data part of the message. The data part of the message is properly formatted if it contains a null-terminated format string, followed by up to three arguments packed one word each after the end of the string.

A different series of sequence numbers is provided for error and trace logging streams. These sequence numbers are intended to help track the delivery of the messages. A gap in a sequence of numbers indicates that the logger process did not successfully deliver them. This can happen if the logger process stops sending messages for one reason or another (see `strace(1M)` and `strerr(1M)` command reference pages for more information). The data part of messages contains text of the format string (null terminated), followed by up to three arguments.

STREAMS-NetTL Link

Both STREAMS error logging and event tracing messages are mapped to NetTL logging messages, and are delivered to NetTL. NetTL classifies messages into four log classes: DISASTER, ERROR, WARNING, and INFORMATIVE. The NetTL log class is determined by the `flags` according to the following rule:

```
If (flags & SL_ERROR)      NetTL log class
then
    if (flags & SL_FATAL) ==> DISASTER
    if (flags & SL_WARN)  ==> WARNING
    if (flags & SL_NOTE)  ==> INFORMATIVE
    otherwise             ==> ERROR
```

```
else
    all messages      ==> INFORMATIVE
```

As a default, only DISASTER and ERROR messages are logged. This setting can be altered by the **nettl** command or the **nettlconf** command (see *nettl(1M)* and *nettlconf(1M)*).

The STREAMS subsystem ID used by NetTL is **STREAMS**.

The messages logged by NetTL facility can be formatted to a readable form by the **netfmt** command (see *netfmt(1M)*). The **netfmt** accepts a filter configuration file, which can be used to filter on STREAMS module ID and sub-ID. The filter configuration file syntax for STREAMS is the following:

```
STREAMS module_id sub_id
```

module_id and *sub_id* can be a decimal number or "*" as a wild card.

RETURN VALUE

Unless specified otherwise, upon successful completion, the **strlog ioctl()** commands return a value of 0 (zero). Otherwise, a value of -1 is returned.

ERRORS

If any of the following conditions occurs, **strlog** driver's **ioctl()** command sets **errno** to the corresponding value:

[ENXIO] The **I_TRCLOG** **ioctl()** call did not contain any **trace_ids** structures.

[ENXIO] The **I_STR** **ioctl()** call could not be recognized.

The driver does not return any errors for incorrectly formatted messages that user processes send.

EXAMPLES

The following examples illustrate some basic uses for the **strlog** interface.

This code example segment shows how a STREAMS module causes a message to be printed to the console:

```
strlog(TMUX,minor(mydev),0,SL_CONSOLE|SL_FATAL,
      "TMUX driver (minor:%d) suffers resource shortage.",
      minor(mydev));
```

This code example shows how a user process registers itself with the STREAMS log driver using the **ioctl()** command, **I_ERRLOG**.

```
struct striocctl iocerr;
int logfd;

if ((logfd = open("/dev/strlog", O_RDWR)) == -1) {
    printf("Cannot open /dev/strlog\n");
    exit(1);
}

iocerr.ic_cmd = I_ERRLOG;
iocerr.ic_timeout = 0;
iocerr.ic_len = 0;
iocerr.ic_dp = NULL;
ioctl(logfd, I_STR, &iocerr);
```

This code example shows a user-level process sending a message to the **strlog** driver.

```
struct strbuf control, data;
struct log_ctl log;
char *warning = "Fatal error for user level process";
int logfd;

if ((logfd = open("/dev/strlog", O_RDWR)) == -1) {
    printf("Cannot open /dev/strlog\n");
    exit(1);
}

control.len = control.maxlen = sizeof(log);
```

```

control.buf = (char *)&lc;

data.len = data.maxlen = strlen(warning);
data.buf = warning;

lc.level = 2;
lc.flags = SL_FATAL|SL_CONSOLE;

putmsg(logfd, &control, &data, 0);

```

The following examples illustrate how to use the NetTL facility for the STREAMS. See *nettl*(1M), *netfmt*(1M), *nettlconf*(1M) for the general NetTL usage. The STREAMS subsystem ID used by NetTL is **STREAMS**.

The **netfmt** accepts a filter configuration file as a command argument. The following filter configuration file example is used to format the messages whose module ID is 1 and sub-ID is 100:

```
STREAMS 1 100
```

This filter configuration file example can be used to display all the messages whose module ID is 2 and all the messages whose sub-ID is 101:

```
STREAMS 2 *
STREAMS * 101
```

FILES

/dev/strlog	specifies the clone interface.
<sys/strlog.h>	specifies the header file for streams logging.
<stropts.h>	specifies the header file for STREAMS options and ioctl() commands.

SEE ALSO

strace(1M), strerr(1M), clone(7), streamio(7), getmsg(2), putmsg(2), write(2), open(2), ioctl(2), nettl(1M), netfmt(1M), nettlconf(1M).

NAME

stty - terminal interface for Version 6/PWB compatibility

REMARKS

These facilities are included to aid in conversion of old programs, and should not be used in new code. Use the interface described in *termio*(7). Note that these conversions do *not* work for programs ported from UNIX Time-Sharing System, Seventh Edition (Version 7), because some V7 flags are defined differently.

DESCRIPTION

These routines attempt to map the UNIX Time-Sharing System, Sixth Edition (Version 6), and PWB *stty* and *gtty* calls into the current *ioctl*s that perform the same functions. The mapping cannot be perfect. The way the features are translated is described below. The reader should be familiar with *termio*(7) before studying this entry.

The following data structure is defined in the include file **<sgtty.h>**:

```
struct sgttyb {
    char  sg_ispeed; /* input speed */
    char  sg_ospeed; /* output speed */
    char  sg_erase;  /* erase character */
    char  sg_kill;   /* kill character */
    int   sg_flags;  /* mode flags */
}
```

The flags, as defined in **sgtty.h**, are:

```
#define HUPCL    01
#define XTABS    02
#define LCASE    04
#define ECHO     010
#define CRMOD    020
#define RAW      040
#define ODDP     0100
#define EVENP    0200
#define ANYP     0300
#define NLDELAY  001400
#define TBDELAY  002000
#define CRDELAY  030000
#define VTDELAY  040000
#define BSDELAY   0100000

#define CR0      0
#define CR1      010000
#define CR2      020000
#define CR3      030000
#define NL0      0
#define NL1      000400
#define NL2      001000
#define NL3      001400
#define TAB0     0
#define TAB1     002000
#define NOAL     004000
#define FF0      0
#define FF1      040000
#define BS0      0
#define BS1      0100000
```

When the *stty*(2) command (*ioctl* **TIOCSETP**) is executed, the flags in the old **sgttyb** structure are mapped into their new equivalents in the **termio** structure. Then the **TCSETA** command is executed.

The following table shows the mapping between the old **sgttyb** flags and the current **termio** flags. Note that flags contained in the **termio** structure that are not mentioned below are cleared.

HUPCL	(if set) sets the termio HUPCL flag;
HUPCL	(if clear) clears the termio HUPCL flag;

XTABS	(if set) sets the termio TAB3 flag;
XTABS	(if clear) clears the termio TAB3 flag;
TBDELAY	(if set) sets the termio TAB1 flag;
TBDELAY	(if clear) clears the termio TAB1 flag;
LCASE	(if set) sets the termio IUCLC, OLCUC, and XCASE flags;
LCASE	(if clear) clears the termio IUCLC, OLCUC, and XCASE flags;
ECHO	(if set) sets the termio ECHO flag;
ECHO	(if clear) clears the termio ECHO flag;
NOAL	(if set) sets the termio ECHOK flag;
NOAL	(if clear) clears the termio ECHOK flag;
CRMOD	(if set) sets the termio ICRNL and ONLCR flags; also, if CR1 is set, the termio CR1 flag is set, and if CR2 is set, the termio ONOCR and CR2 flags are set;
CRMOD	(if clear) sets the termio ONLRET flag; also, if NL1 is set, the termio CR1 flag is set, and if NL2 is set, the termio CR2 flag is set;
RAW	(if set) sets the termio CS8 flag, and clears the termio ICRNL and IUCLC flags; also, default values of 6 characters and 0.1 seconds are assigned to MIN and TIME, respectively;
RAW	(if clear) sets the termio BRKINT, IGNPAR, ISTRIP, IXON, IXANY, OPOST, CS7, PARENB, ICANON, and ISIG flags; also, the default values control-D and null are assigned to the control characters EOF and EOL, respectively;
ODDP	(if set) if EVENP is also set, clears the termio INPCK flag; otherwise, sets the termio PARODD flag;
VTDELAY	(if set) sets the termio FFDLY flag;
VTDELAY	(if clear) clears the termio FFDLY flag;
BSDLY	(if set) sets the termio BSDLY flag;
BSDLY	(if clear) clears the termio BSDLY flag.

In addition, the **termio** CREAD bit is set, and, if the baud rate is 110, the CSTOPB bit is set.

When using **TIOCSETP**, the *ispeed* entry in the **sgttyb** structure is mapped into the appropriate speed in the **termio** CBAUD field. The *erase* and *kill* **sgttyb** entries are mapped into the **termio** erase and kill characters.

When the *gtty(2)* (*ioctl* **TIOCGETP**) command is executed, the *termio(7)* **TCGETA** command is first executed. The resulting **termio** structure is then mapped into the **sgttyb** structure, which is then returned to the user.

The following table shows how the **termio** flags are mapped into the old **sgttyb** structure. Note that all flags contained in the **sgttyb** structure that are not mentioned below are cleared.

HUPCL	(if set) sets the sgttyb HUPCL flag;
HUPCL	(if clear) clears the sgttyb HUPCL flag;
ICANON	(if set) sets the sgttyb RAW flag;
ICANON	(if clear) clears the sgttyb RAW flag;
XCASE	(if set) sets the sgttyb LCASE flag;
XCASE	(if clear) clears the sgttyb LCASE flag;
ECHO	(if set) sets the sgttyb ECHO flag;
ECHO	(if clear) clears the sgttyb ECHO flag;
ECHOK	(if set) sets the sgttyb NOAL flag;
ECHOK	(if clear) clears the sgttyb NOAL flag;
PARODD	(if set) sets the sgttyb ODDP flag;
PARODD	(if clear) clears the sgttyb ODDP flag;
INPCK	(if set) sets the sgttyb EVENP flag;
PARODD,	INPCK (if both clear) sets the sgttyb ODDP and EVENP flags;
ONLCR	(if set) sets the sgttyb CRMOD flag; also, if CR1 is set, the sgttyb CR1 flag is set, and if CR2 is set, the sgttyb CR2 flag is set;
ONLCR	(if clear) if CR1 is set, the sgttyb NL1 flag is set, and if CR2 is set, the sgttyb NL2 flag is set;
TAB3	(if set) sets the sgttyb XTABS flag;
TAB3	(if clear) clears the sgttyb XTABS flag;
TAB1	(if set) sets the sgttyb TBDELAY flag;
TAB1	(if clear) clears the sgttyb TBDELAY flag;
FFDLY	(if set) sets the sgttyb VTDELAY flag;
FFDLY	(if clear) clears the sgttyb VTDELAY flag;
BSDLY	(if set) sets the sgttyb BSDLY flag;
BSDLY	(if clear) clears the sgttyb BSDLY flag.

When using **TIOCGETP**, the **termio** CBAUD field is mapped into the *ispeed* and *ospeed* entries of the **sgttyb** structure. Also, the **termio** erase and kill characters are mapped into the *erase* and *kill* **sgttyb** entries.

Note that, since there is not a one-to-one mapping between the **sgttyb** and **termio** structures, unexpected results may occur when using the older **TIOCSETP** and **TIOCGETP** calls. Thus, the **TIOCSETP** and **TIOCGETP** calls should be replaced in all future code by the current equivalents, **TCSETA** and **TCGETA**, respectively.

SEE ALSO

stty(2), termio(7).

NAME

TCP - Internet Transmission Control Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>

s = socket(AF_INET, SOCK_STREAM, 0);
```

DESCRIPTION

The TCP protocol provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the `SOCK_STREAM` socket type. TCP constructs virtual circuits between peer entities. A virtual circuit consists of remote Internet addresses, remote ports, local Internet addresses and local ports. IP uses the Internet addresses to direct messages between hosts, and the port numbers to identify a TCP entity at a particular host.

Sockets using TCP are either **active** or **passive**. `connect()` creates active sockets, which initiate connections to passive sockets (see `connect(2)`). To create a passive socket, use the `listen()` system call after binding the socket with the `bind()` system call (see `listen(2)` and `bind(2)`). Only passive sockets can use the `accept()` call to accept incoming connections (see `accept(2)`).

Passive sockets can *underspecify* their location to match incoming connection requests from multiple networks. This technique, called **wildcard addressing**, allows a single server to provide service to clients on multiple networks. To create a socket that listens on all networks, the Internet address `INADDR_ANY` must be bound. The TCP port can still be specified even if wildcard addressing is being used. If the port is specified as zero, the system assigns a port.

Once `accept()` has a rendezvous with a connect request, a virtual circuit is established between peer entities. `bind()` supplies the local port and local Internet address and `accept()` gathers the remote port and remote Internet address from the peer requesting the connection.

The system supports four socket options: `TCP_MAXSEG`, `TCP_NODELAY`, `TCP_ABORT_THRESHOLD`, and `TCP_CONN_ABORT_THRESHOLD` (defined in the include file `<netinet/tcp.h>`). `TCP_MAXSEG` option can only be used with `getsockopt()`, while `TCP_NODELAY`, `TCP_ABORT_THRESHOLD`, and `TCP_CONN_ABORT_THRESHOLD` can be set with `setsockopt()` and tested with `getsockopt()` (see `getsockopt(2)`). These four options require *level* to be set to `IPPROTO_TCP` in the `getsockopt/setsockopt` call.

TCP_MAXSEG (non-boolean option) lets an application to receive the current segment size of the TCP `SOCK_STREAM` socket. The current segment size will be returned in optval.

TCP_NODELAY (boolean option) causes small amounts of output to be sent immediately.

TCP_ABORT_THRESHOLD (non-boolean option) sets the second threshold timer for the connections that are in ESTABLISHED state. The option value is the threshold time in milliseconds.

When it must retransmit packets because a timer has expired, TCP first compares the total time it has waited against the two thresholds, as described in RFC 1122, 4.2.3.5. If it has waited longer than the second threshold (R2), TCP terminates the connection. The default value for this option is the current value of the ndd tunable parameter `tcp_ip_abort_interval`. Refer to `ndd(1M)` online help for details on the `tcp_ip_abort_interval` default value.

TCP_CONN_ABORT_THRESHOLD (non-boolean option) sets the second threshold timer during connection establishment. The option value is the threshold time in milliseconds.

This option is the same as `TCP_ABORT_THRESHOLD`, except that this value is used during connection establishment. When it must retransmit the SYN packet because a timer has expired, TCP first compares the total time it has waited against the two thresholds. If it has waited longer than the second threshold, TCP terminates the connection. The default value for this option is the current value of the ndd tunable

`tcp_ip_abort_cinterval`. See *ndd(1M)* online help for details on the `tcp_ip_abort_cinterval` default value.

If `TCP_NODELAY` is set, the system sends small amounts of output immediately rather than gathering them into a single packet after an acknowledgement is received. If `TCP_NODELAY` is not set, the system sends data when it is presented, if there is no outstanding unacknowledged data. If there is outstanding unacknowledged data, the system gathers small amounts of data to be sent in a single packet once an acknowledgement is received. For clients such as window managers that send a stream of mouse events which receive no replies, this packetization may cause significant delays. The `TCP_NODELAY` option can be used to avoid this situation. Note, however, that setting the `TCP_NODELAY` option may result in a large number of small packets being sent over the network.

By default, `TCP_NODELAY` is not set when a socket is created.

The option level to use for accessing the TCP option with the `setsockopt()` or `getsockopt()` calls is the protocol number for TCP which is available from `getprotobyname()` (see *getprotoent(3N)*).

If the `SO_KEEPAIVE` socket option is enabled on an established TCP connection and the connection has been idle for two hours, TCP sends a packet to the remote socket, expecting the remote TCP to acknowledge that the connection is still active. If the remote TCP does not respond in a timely manner, TCP continues to send keepalive packets according to its normal retransmission algorithm. If the remote TCP does not respond within a particular time limit, TCP drops the connection. The next socket system call (for example, `recv()`) returns an error, and `errno` is set to `ETIMEDOUT`. See *getsockopt(2)* for details on enabling `SO_KEEPAIVE`.

The maximum buffer size for a TCP stream socket is 262142 bytes. The default buffer size is 32768 bytes (see *WARNINGS* below). The send and receive buffer sizes for TCP stream sockets can be altered by using the `SO_SNDBUF` and `SO_RCVBUF` options of the `setsockopt()` system call. Refer to *getsockopt(2)* for details.

ERRORS

One of the following errors may be returned in `errno` if a socket operation fails. For a more detailed list of errors, see the man pages for specific system calls.

<code>[EISCONN]</code>	The socket is already connected.
<code>[ENOBUFS]</code>	No buffer space is available for an internal data structure.
<code>[ETIMEDOUT]</code>	Connection dropped due to excessive retransmissions.
<code>[ECONNRESET]</code>	The connection was forcibly closed by the peer socket.
<code>[ECONNREFUSED]</code>	Remote peer actively refuses connection establishment (usually because no process is listening to the port).
<code>[EADDRINUSE]</code>	The specified address is already in use.
<code>[EADDRNOTAVAIL]</code>	The specified address is not available on this machine.

WARNINGS

The default socket buffer size might increase without notice in a future release or patch. Therefore, if an application calls `setsockopt()` with `SO_RCVBUF`, it should do so before calling `listen()`, or it should first call `getsockopt()` with `SO_RCVBUF` and ensure that the intended new receive buffer size is not less than the current buffer size. These programming conventions are consistent with TCP protocol restrictions against reducing the TCP receive window after a connection has been established.

AUTHOR

The socket interfaces to TCP were developed by the University of California, Berkeley.

SEE ALSO

getsockopt(2), *socket(2)*, *send(2)*, *recv(2)*, *t_open(3)*, *socket(7)*, *inet(7F)*, *ndd(1M)*.

RFC 793 Transmission Control Protocol
RFC 1122 Requirements for Internet hosts

NAME

tels, telm - STREAMS Telnet slave (pseudo-terminal) driver, STREAMS Telnet master driver (used by telnetd only), respectively

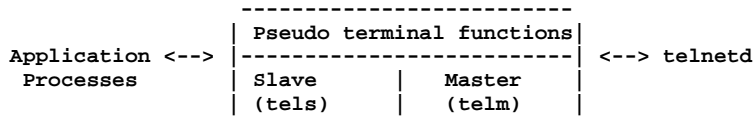
SYNOPSIS

```
#include <sys/termios.h>
#include <sys/strtio.h>

int open("/dev/pts/tN", O_RDWR);
```

DESCRIPTION

A Telnet pseudo-terminal consists of a tightly-coupled pair of character devices, called the master device and slave device. The master and slave device drivers work together to provide a Telnet connection on the server side where the master provides a connection to **telnetd** and the slave provides a terminal device special file access for the Telnet application processes, as depicted below:



The slave driver, **tels** with **ptem** (STREAMS pty emulation module) and **ldterm** (STREAMS line discipline module) pushed on top (not shown for simplicity), provides a terminal interface as described in *termio(7)*. Whereas devices that provide the terminal interface described in *termio(7)* have a hardware device behind them; in contrast, the slave device has **telnetd** manipulating it through the master side of the Telnet pseudo terminal.

There are no nodes in the file system for each individual master device. Rather, the master driver is set up as a STREAMS *clone(7)* driver with its major device number set to the major for the clone driver and its minor device number set to the major for the **telm** driver. The master driver is opened by **telnetd** using the *open(2)* system call with **/dev/telnetm** as the device file parameter. The clone open finds the next available minor number for the master device. The master device is available only if it and its corresponding slave device are not already opened.

In order to use the STREAMS Telnet subsystem, a node for the master driver **/dev/telnetm** and *N* number of Telnet slave devices must be installed.

The number of slave devices is set by a kernel tunable parameter called **nstrtel**. This can be modified using SAM; its default and minimum value is 60. The value of **nstrtel** is the upper limit of the number of telnet sessions that can be opened.

Multiple opens are allowed on the Telnet slave device.

The master and slave drivers pass all STREAMS messages to their adjacent drivers. When the connection is closed from the Telnet client side, an **M_HANGUP** message is sent to the corresponding slave device which will render that slave device unusable. The process on the slave side gets the errno **ENXIO** when attempting a *write(2)* system call to the slave device file but it will be able to read any data remaining in the slave stream. Finally, when all the data has been read, the *read(2)* system call will return 0, indicating that the slave can no longer be used.

AUTHOR

tels() and **telm()** were developed by HP.

FILES

/dev/telnetm	Streams Telnet master clone device
/dev/pts/tN	Streams slave devices where <i>N</i> is the minor number of the slave device and $0 < N < \text{nstrtel}$.

SEE ALSO

insf(1M), **open(2)**, **ioctl(2)**, **streamio(7)**, **ldterm(7)**, **telnetd(1M)**, **ptem(7)**.

NAME

termio, termios - general terminal interface

DESCRIPTION

All HP-UX asynchronous communications ports use the same general interface, regardless of what hardware is involved. Network connections such as **rlogin** (see *rlogin(1)*) use the pseudo-terminal interface (see *pty(7)*).

This discussion centers around the common features of this interface.

Opening a Terminal File

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open these files; they are opened by special programs such as **getty** (see *getty(1M)*) and become a user's standard input, standard output, and standard error files.

If both the **O_NDELAY** and **O_NONBLOCK** flags (see *open(2)*) are clear, an *open* blocks until the type of modem connection requested (see *modem(7)*) is completed. If either the **O_NDELAY** or **O_NONBLOCK** flag is set, an *open* succeeds and return immediately without waiting for the requested modem connection to complete. The **CLOCAL** flag (see *Control Modes*) can also affect *open(2)*.

Process Groups

A terminal can have a foreground process group associated with it. This foreground process group plays a special role in handling signal-generating input characters.

Command interpreter processes can allocate the terminal to different *jobs* (process groups) by placing related processes in a single process group and associating this process group with the terminal. A terminal's foreground process group can be set or examined by a process, assuming that the permission requirements are met (see *tcsetpgrp(3C)* or *tcgetpgrp(3C)*). The terminal interface aids in this allocation by restricting access to the terminal by processes that are not in the foreground process group.

A process group is considered orphaned when the parent of every member of the process group is either itself a member of the process group or is not a member of the group's session (see *Sessions*).

Sessions

A process that creates a session (see *setsid(2)* or *setpgrp(2)*) becomes a session leader. Every process group belongs to exactly one session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its parent. A process can change its session membership (see *setpgid(2)* or *setpgrp(2)*). Usually a session comprises all the processes (including children) created as a result of a single login.

The Controlling Terminal

A terminal can belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal can be the controlling terminal for at most one session. The controlling terminal for a session is allocated by the session leader. If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the **O_NOCTTY** option (see *open(2)*), the terminal becomes the controlling terminal of the session and the controlling terminal's foreground process group is set to the process group of the session leader. While a controlling terminal is associated with a session, the session leader is said to be the controlling process of the controlling terminal.

The controlling terminal is inherited by a child process during a **fork()** (see *fork(2)*). A process relinquishes its controlling terminal if it creates a new session with **setsid()** or **setpgrp()** (see *setsid(2)* and *setpgrp(2)*), or when all file descriptors associated with the controlling terminal have been closed.

When the controlling process terminates, the controlling terminal is disassociated from the current session, allowing it to be acquired by a new session leader. A **SIGHUP** signal is sent to all processes in the foreground process group of the controlling terminal. Subsequent access to the terminal by other processes in the earlier session can be denied (see *Terminal Access Control*) with attempts to access the terminal treated as if a modem disconnect had been sensed.

Terminal Access Control

Read operations are allowed (see *Input Processing and Reading Data*) from processes in the foreground process group of their controlling terminal. If a process is not in the foreground process group of its controlling terminal, the process and all member's of its process group are considered to be in a background process group of this controlling terminal. All attempts by a process in a background process group to read

from its controlling terminal will be denied. If denied and the reading process is ignoring or blocking the **SIGTTIN** signal, or the process (on systems that implement *vfork* separately from *fork*) has made a call to *vfork*(2) but has not yet made a call to *exec*(2), or the process group of the reading process is orphaned, *read*() returns -1 with **errno** set to **EIO** and no signal is sent. In all other cases where the read is denied, the process group of the reading process will be sent a **SIGTTIN** signal. The default action of the **SIGTTIN** signal is to stop the process to which it is sent.

If the process is in the foreground process group of its controlling terminal, write operations are allowed (see *Writing Data and Output Processing*). Attempts by a process in a background process group to write to its controlling terminal are denied if **TOSTOP** (see *Local Modes*) is set, the process is not ignoring and not blocking the **SIGTTTOU** signal, and the process (on systems that implement *vfork* separately from *fork*) has not made a call to *vfork*(2) without making a subsequent call to *exec*(2). If the write is denied and the background process group is orphaned, the *write*() returns -1 with **errno** set to **EIO**. If the write is denied and the background process group is not orphaned, the **SIGTTTOU** signal is sent to the process group of the writing process. The default action of the **SIGTTTOU** signal is to stop the process to which it is sent.

Certain calls that set terminal parameters are treated in the same fashion as write, except that **TOSTOP** is ignored; that is, the effect is identical to that of terminal writes when **TOSTOP** is set.

Input Processing and Reading Data

A terminal device associated with a terminal device file can operate in full-duplex mode, so that data can arrive, even while data output is occurring. Each terminal device file has an *input queue* associated with it into which incoming data is stored by the system before being read by a process. The system imposes a limit, **MAX_INPUT**, on the number of characters that can be stored in the input queue. This limit is dependent on the particular implementation, but is at least 256. When the input limit is reached, all saved characters are discarded without notice.

All input is processed either in canonical mode or non-canonical mode (see *Canonical Mode Input Processing* and *Non-Canonical Mode Input Processing*). Additionally, input characters are processed according to the **c_iflag** (see *Input Modes*) and **c_lflag** (see *Local Modes*) fields. For example, such processing can include *echoing*, which in general means transmitting input characters immediately back to the terminal when they are received from the terminal. This is useful for terminals that operate in full-duplex mode.

The manner in which data is provided to a process reading from a terminal device file depends on whether the terminal device file is in canonical or non-canonical mode.

Another dependency is whether the **O_NONBLOCK** or **O_NDELAY** flag is set by either *open*(2) or *fcntl*(2). If the **O_NONBLOCK** and **O_NDELAY** flags are both clear, the read request is blocked until data is available or a signal is received. If either the **O_NONBLOCK** or **O_NDELAY** flag is set, the read request completes without blocking in one of three ways:

- If there is enough data available to satisfy the entire request, *read*() completes successfully, having read all of the data requested, and returns the number of characters read.
- If there is not enough data available to satisfy the entire request, *read*() completes successfully, having read as much data as possible, and returns the number of characters read.
- If there is no data available, *read*() returns -1, with **errno** set to **EAGAIN** when the **O_NONBLOCK** flag is set. Otherwise, (flag **O_NONBLOCK** is clear and **O_NDELAY** is set) *read*() completes successfully, having read no data, and returns a count of 0.

The availability of data depends upon whether the input processing mode is canonical or non-canonical. The following sections, *Canonical Mode Input Processing* and *Non-Canonical Mode Input Processing*, describe each of these input processing modes.

Canonical Mode Input Processing (Erase and Kill Processing)

In canonical mode input processing, terminal input is processed in units of lines, where a line is delimited by a new-line (NL) character, an end-of-file (EOF) character, or an end-of-line character (EOL) or (EOL2). See *Special Characters* for more information on **NL**, **EOF**, **EOL**, and **EOL2**. This means that a read request does not return until an entire line has been typed or a signal has been received. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters can be requested in a read, even one, without losing information.

MAX_CANON is the limit on the number of characters in a line. This limit varies with each particular implementation, but is at least 256.

When the **MAX_CANON** limit is reached, all characters in the current un delimited line are discarded without notice.

Erase and kill processing occur when any of three special characters, the ERASE, WERASE, or KILL characters (see *Special Characters*), is received. This processing affects data in the input queue that has not yet been delimited by a NL, EOF, EOL, or EOL2 character. This un delimited data makes up the current line. The ERASE character deletes the last character in the current line, if one exists. The WERASE character deletes the last word in the current line, if one exists. A word is defined as a series of non-blank characters (tabs are equivalent to blanks). The KILL character deletes all data in the current line, if any, and optionally outputs a new-line (NL) character. These characters operate on a key-stroke basis, independent of any backspacing or tabbing that may have preceded them. ERASE, WERASE, and KILL characters have no effect if the current line is empty. ERASE, WERASE, and KILL characters are not placed in the input queue.

Non-Canonical Mode Input Processing (MIN/TIME Interaction)

In non-canonical mode input processing, input characters are not assembled into lines, and erase and kill processing does not occur. The values of the **MIN** and **TIME** members of the **c_cc** array (see *termios Structure*) are used to determine how to process the characters received. **MIN** represents the minimum number of characters that should be received before **read()** successfully returns. **TIME** is a timer of 0.10 second granularity that is used to timeout bursty and short term data transmissions. The four possible cases for **MIN** and **TIME** and their interactions are described below.

Case A: **MIN** > 0, **TIME** > 0

In this case, **TIME** serves as an inter-character timer and is activated after the first character is received. Since it is an inter-character timer, it is reset after each character is received. The interaction between **MIN** and **TIME** is as follows:

- As soon as one character is received, the inter-character timer is started.
- If **MIN** characters are received before the inter-character timer expires (remember that the timer is reset upon receipt of each character), the read is satisfied. If the timer expires before **MIN** characters are received, the characters received to that point are returned to the user.
- Note that if **TIME** expires, at least one character will be returned because the timer would not have been enabled unless a character was received. In this case (**MIN** > 0, **TIME** > 0) the read blocks until the **MIN** and **TIME** mechanisms are activated by the receipt of the first character, or a signal is received.

Case B: **MIN** > 0, **TIME** = 0

In this case, since the value of **TIME** is zero, the timer plays no role and only **MIN** is significant. A pending read is not satisfied until **MIN** characters are received after any previous read completes (that is, the pending read blocks until **MIN** characters are received), or a signal is received. A program that uses this case to handle record-based terminal I/O can block indefinitely in the read operation.

Case C: **MIN** = 0, **TIME** > 0

In this case, since the value of **MIN** is zero, **TIME** no longer represents an inter-character timer. It now serves as a read timer that is activated as soon as the **read()** function is processed. A read is satisfied as soon as a single character is received *or* the read timer expires. If the timer expires, no character is returned. If the timer does not expire, the only way the read can be satisfied is by a character being received. A read cannot block indefinitely waiting for a character because if no character is received within **TIME** × 0.10 seconds after the read is initiated, **read()** returns a value of zero, having read no data.

Case D: **MIN** = 0, **TIME** = 0

The number of characters requested or the number of characters currently available, whichever is less, is returned without waiting for more characters to be input. If no characters are available, **read()** returns a value of zero, having read no data.

Some points to note about **MIN** and **TIME**:

1. In the above explanations, the interactions of **MIN** and **TIME** are not symmetric. For example, when **MIN** > 0 and **TIME** = 0, **TIME** has no effect. However, in the opposite case where **MIN** = 0 and **TIME** > 0, both **MIN** and **TIME** play a role in that **MIN** is satisfied with the receipt of a single character.
2. Also note that in case A (**MIN** > 0, **TIME** > 0), **TIME** represents an inter-character timer while in case C (**MIN** = 0, **TIME** > 0), **TIME** represents a read timer.

These two points highlight the dual purpose of the MIN/TIME feature. Cases A and B (where MIN > 0) exist to handle burst mode activity (such as file transfer programs) where a program would like to process at least MIN characters at a time. In case A, the inter-character timer is activated by a user as a safety measure while in case B it is turned off.

Cases C and D exist to handle single character timed transfers. These cases are readily adaptable to screen-based applications that need to know if a character is present in the input queue before refreshing the screen. In case C the read is timed, while in case D it is not.

Another important note is that MIN is always just a minimum. It does not denote a record length. For example, if a program initiates a read of 20 characters when MIN is 10 and 25 characters are present, 20 characters will be returned to the user. Had the program requested all characters, all 25 characters would be returned to the user.

Furthermore, if TIME is greater than zero and MIN is greater than **MAX_INPUT**, the read will never terminate as a result of MIN characters being received because all the saved characters are discarded without notice when **MAX_INPUT** is exceeded. If TIME is zero and MIN is greater than **MAX_INPUT**, the read will never terminate unless a signal is received.

Special Characters

Certain characters have special functions on input, output, or both. Unless specifically denied, each special character can be changed or disabled. To disable a character, set its value to **_POSIX_VDISABLE** (see *unistd(5)*). These special functions and their default character values are:

INTR	(Rubout or ASCII DEL) special character on input and is recognized if ISIG (see <i>Local Modes</i>) is enabled. Generates a SIGINT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. Normally, each such process is forced to terminate, but arrangements can be made to either ignore or hold the signal, or to receive a trap to an agreed-upon location; see <i>signal(2)</i> and <i>signal(5)</i> . If ISIG is set, the INTR character is discarded when processed. If ISIG is clear, the INTR character is processed as a normal data character, and no signal is sent.
QUIT	(Ctrl- or ASCII FS) special character on input. Recognized if ISIG (see <i>Local Modes</i>) is set. The treatment of this character is identical to that of the INTR character except that a SIGQUIT signal is generated and the processes that receive this signal are not only terminated, but a core image file (called core) is created in the current working directory if the implementation supports core files.
SWTCH	(ASCII NUL) special character on input and is only used by the shell layers facility <i>shl(1)</i> . The shell layers facility is not part of the general terminal interface. No special functions are performed by the general terminal interface when SWTCH characters are encountered.
ERASE	(#) special character on input and is recognized if ICANON (see <i>Local Modes</i>) is enabled. Erases the preceding character. Does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the ERASE character is discarded when processed. If ICANON is not enabled, the ERASE character is treated as a normal data character.
WERASE	(disabled) special character on input and is recognized if ICANON (see <i>Local Modes</i>) is enabled. Erases the preceding word. Does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the WERASE character is discarded when processed. If ICANON is not enabled, the WERASE character is treated as a normal data character.
KILL	(@) special character on input and is recognized if ICANON is enabled. KILL deletes the entire line, as delimited by a NL, EOF, EOL, or EOL2 character. If ICANON is enabled, the KILL character is discarded when processed. If ICANON is not enabled, the KILL character is treated as a normal data character.
EOF	(Control-D or ASCII EOT) special character on input and is recognized if ICANON is enabled. EOF can be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, (that is, the EOF occurred at the beginning of a line) a character count of zero is returned from read() , representing an end-of-file indication. If ICANON is enabled, the EOF character is discarded when processed. If ICANON is not enabled, the EOF

character is treated as a normal data character.

NL	(ASCII LF) special character on input and is recognized if ICANON flag is enabled. It is the line delimiter (\n). If ICANON is not enabled, the NL character is treated as a normal data character.
EOL	(ASCII NUL) special character on input and is recognized if ICANON is enabled. EOL is an additional line delimiter similar to NL. It is not normally used. If ICANON is not enabled, the EOL character is treated as a normal data character.
EOL2	(disabled) special character on input and is recognized if ICANON is enabled. EOL2 is an additional line delimiter similar to EOL. It is not normally used. If ICANON is not enabled, the EOL2 character is treated as a normal data character.
SUSP	(disabled) special character recognized on input. If ISIG is enabled, receipt of the SUSP character causes a SIGTSTP signal to be sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the SUSP character is discarded when processed. If ISIG is not enabled, the SUSP character is treated as a normal data character. Command interpreter processes typically set SUSP to Control-Z.
DSUSP	(disabled) special character recognized on input. If ISIG is enabled, and a process in the foreground process group attempts to read the DSUSP character, a SIGTSTP signal is sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the DSUSP character is then discarded. If ISIG is not enabled, the DSUSP character is treated as a normal data character. Note that DSUSP is similar to SUSP except that the signal is sent when a process in the foreground process group attempts to read the DSUSP character, rather than when it is typed.
STOP	(Control-S or ASCII DC3) special character on both input and output. If IXON (output control) is enabled, processing of the STOP character temporarily suspends output to the terminal device. This is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended and IXON is enabled, STOP characters are ignored and not read. If IXON is enabled, the STOP character is discarded when processed. If IXON is not enabled, the STOP character is treated as a normal data character. If IXOFF (input control) is enabled, the system sends a STOP character to the terminal device when the number of unread characters in the input queue is approaching a system specified limit. This is an attempt to prevent this buffer from overflowing by telling the terminal device to stop sending data.
START	(Control-Q or ASCII DC1) special character on both input and output. If IXON (output control) is enabled, processing of the START character resumes output that has been suspended. While output is not suspended and IXON is enabled, START characters are ignored and not read. If IXON is enabled, the START character is discarded when processed. If IXON is not enabled, the START character is treated as a normal data character. If IXOFF (input control) is enabled, the system sends a START character to the terminal device when the input queue has drained to a certain system-defined level. This occurs when the input queue is no longer in danger of possibly overflowing.
CR	(ASCII CR) special character on input is recognized if ICANON is enabled. When ICANON and ICRNL are enabled and IGNCR is not enabled, this character is translated into a NL, and has the same affect as the NL character. If ICANON and IGNCR are enabled, the CR character is ignored. If ICANON is enabled and both ICRNL and IGNCR are not enabled, the CR character is treated as a normal data character.
LNEXT	(disabled) special character recognized on input. Causes the special meaning of the next character to be ignored. This works for all special characters specified above. It allows characters to be input that would otherwise be interpreted by the system for a special function.

The special characters are assigned their default character values when the terminal port is opened. The default values used are those specified by the System V Interface Definition, Third Edition (SVID3), except for the WERASE (Ctrl-W) and LNEXT (Ctrl-V) characters which are set to **_POSIX_VDISABLE** to maintain binary compatibility with previous releases of HP-UX. The default character values assigned when the port is opened can be changed for all ports on a system wide basis through the use of the **stty** command (see *stty(1)*). The character values may also be changed for a specific port after it is opened using the **stty** command. The NL and CR characters cannot be changed or disabled. The character values for the

remaining special characters can be changed or disabled to suit individual tastes.

If **ICANON** is set (see *Local Modes*), the ERASE, KILL, and EOF characters can be escaped by a preceding **** character, in which case no special function is performed. These characters, and the remaining special characters, may also be escaped by preceding them with the **LNEXT** character (see **LNEXT** above).

If two or more special characters have the same value, the function performed when the character is processed is undefined.

Modem Disconnect

If a modem disconnect is detected by the terminal interface for a controlling terminal, and if **CLOCAL** is clear in the **c_cflag** field for the terminal (see *Control Modes*), the **SIGHUP** signal is sent to the controlling process of the controlling terminal. Unless other arrangements have been made, this causes the controlling process to terminate. Any subsequent read from the terminal device returns with an end-of-file indication until the device is closed. Thus, processes that read a terminal file and test for end-of-file can terminate appropriately after a disconnect. Any subsequent **write()** to the terminal device returns **-1**, with **errno** set to **EIO**, until the device is closed.

Closing a Terminal Device File

The last process to close a terminal device file causes any output not already sent to the device to be sent to the device even if output was suspended. This last close always blocks (even if non-blocking I/O has been specified) until all output has been sent to the terminal device. Any input that has been received but not read is discarded.

Writing Data and Output Processing

When characters are written, they are placed on the output queue. Characters on the output queue are transmitted to the terminal as soon as previously-written characters are sent. These characters are processed according to the **c_oflag** field (see *Output Modes*). Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters for output more rapidly than they can be sent, the process is suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the process is resumed.

termios Structure

Routines that need to control certain terminal I/O characteristics can do so by using the **termios** structure as defined in the header file **<termios.h>**. The structure is defined as follows:

```
#define NCCS      16
struct termios {
    tcflag_t  c_iflag;    /* input modes */
    tcflag_t  c_oflag;    /* output modes */
    tcflag_t  c_cflag;    /* control modes */
    tcflag_t  c_lflag;    /* local modes */
    tcflag_t  c_reserved; /* reserved for future use */
    cc_t      c_cc[NCCS]; /* control chars */
};
```

The special characters are defined by the array **c_cc**. The relative positions and default values for each special character function are as follows:

INTR	VINTR	DEL
QUIT	VQUIT	Control-
ERASE	VERASE	#
KILL	VKILL	@
EOF	VEOF	Control-D
EOL	VEOL	NUL
EOL2	VEOL2	disabled
MIN	VMIN	NUL
TIME	VTIME	Control-D
SUSP	VSUSP	disabled
START	VSTART	Control-Q
STOP	VSTOP	Control-S
WERASE	VWERASE	disabled
LNEXT	VLNEXT	disabled
DSUSP	VDSUSP	disabled

termio Structure

The **termio** structure has been superseded by the **termios** structure and is provided for backward compatibility with prior applications (see *termio Caveats*). The structure is defined in the header file `<termio.h>` and is defined as follows:

```
#define NCC      8
struct termio {
    unsigned short  c_iflag;    /* input modes */
    unsigned short  c_oflag;    /* output modes */
    unsigned short  c_cflag;    /* control modes */
    unsigned short  c_lflag;    /* local modes */
    char            c_line;     /* line discipline */
    unsigned char   c_cc[NCC];  /* control chars */
};
```

Modes

The next four sections describe the specific terminal characteristics that can be set using the **termios** and **termio** structures (see *termio Caveats*). Any bits in the modes fields that are not explicitly defined below are ignored. However, they should always be clear to prevent future compatibility problems.

Input Modes

The **c_iflag** field describes the basic terminal input control:

IGNBRK	Ignore break condition.
BRKINT	Signal interrupt on break.
IGNPAR	Ignore characters with parity errors.
PARMRK	Mark parity errors.
INPCK	Enable input parity check.
ISTRIP	Strip character.
INLCR	Map NL to CR on input.
IGNCR	Ignore CR.
ICRNL	Map CR to NL on input.
IUCLC	Map uppercase to lowercase on input.
IXON	Enable start/stop output control.
IXANY	Enable any character to restart output.
IXOFF	Enable start/stop input control.
IMAXBEL	Enable BEL on input line too long.

A break condition is defined as a sequence of zero-value bits that continues for more than the time to send one character. For example, a character framing or parity error with data all zeros is interpreted as a single break condition.

If **IGNBRK** is set, the break condition is ignored. Therefore the break condition cannot be read by any process. If **IGNBRK** is clear and **BRKINT** is set, the break condition flushes both the input and output queues and, if the terminal is the controlling terminal of a foreground process group, the break condition generates a single **SIGINT** signal to that foreground process group. If neither **IGNBRK** nor **BRKINT** is set, a break condition is read as a single `\0` character, or if **PARMRK** is set, as the three-character sequence `\377, \0, \0`.

If **IGNPAR** is set, characters with other framing and parity errors (other than break) are ignored.

If **PARMRK** is set, and **IGNPAR** is clear, a character with a framing or parity error (other than break) is read as the three-character sequence: `\377, \0, X`, where *X* is the data of the character received in error. To avoid ambiguity in this case, if **ISTRIP** is clear, a valid character of `\377` is read as `\377, \377`. If both **PARMRK** and **IGNPAR** are clear, a framing or parity error (other than break) is read as the character `\0`.

If **INPCK** is set, input parity checking is enabled. If **INPCK** is clear, input parity checking is disabled. Whether input parity checking is enabled or disabled is independent of whether parity detection is enabled or disabled (see *Control Modes*). If **PARENB** is set (see *Control Modes*) and **INPCK** is clear, parity generation is enabled but input parity checking is disabled; the hardware to which the terminal is connected will recognize the parity bit, but the terminal special file will not check whether this bit is set correctly or not.

The following table shows the interrelationship between the flags **IGNBRK**, **BRKINT**, **IGNPAR**, and **PARMRK**. The column marked **Input** gives various types of input characters received, indicated as follows:

O	NUL character (\0)
C	Character other than NUL
P	Parity error detected
F	Framing error detected

Items enclosed in brackets indicate one or more of the conditions are true.

If the **INPCK** flag is clear, characters received with parity errors are not processed according to this table, but instead, as if no parity error had occurred. Under the flag columns, **Set** indicates the flag is set, **Clear** indicates the flag is not set, and **X** indicates the flag may be set or clear. The column labeled **Read** shows the results that will be passed to the application code. A **—** indicates that no character or condition is passed to the application code. The value **SIGINT** indicates that no character is returned, but that the **SIGINT** signal is sent to the foreground process group of the controlling terminal.

Input	IGNBRK	BRKINT	IGNPAR	PARMRK	Read
0[PF]	Set	X	X	X	—
0[PF]	Clear	Set	X	X	SIGINT
0[PF]	Clear	Clear	X	Set	'\377','\0','\0'
0[PF]	Clear	Clear	X	Clear	'\0'
C[PF]	X	X	Set	X	—
C[PF]	X	X	Clear	Set	'\377','\0',C
C[PF]	X	X	Clear	Clear	'\0'
'\377'	X	X	X	Set	'\377','\377'

If **ISTRIP** is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If **INLCR** is set, a received NL character is translated into a CR character. If **IGNCR** is set, a received CR character is ignored (not read). If **IGNCR** is clear and **ICRNL** is set, a received CR character is translated into a NL character.

If **IUCLC** is set, a received uppercase alphabetic character is translated into the corresponding lowercase character.

If **IXON** is set, start/stop output control is enabled. A received STOP character suspends output and a received START character restarts output. If **IXANY** and **IXON** are set, any input character without a framing or parity error restarts output that has been suspended. When these three flags are set, output suspended, and an input character received with a framing or parity error, output resumes if processing it results in data being read. When **IXON** is set, START and STOP characters are not read, but merely perform flow control functions. When **IXON** is clear, the START and STOP characters are read.

If **IXOFF** is set, start/stop input control is enabled. The system transmits a STOP character when the number of characters in the input queue exceeds a system defined value (high water mark). This is intended to cause the terminal device to stop transmitting data in order to prevent the number of characters in the input queue from exceeding **MAX_INPUT**. When enough characters have been read from the input queue that the number of characters remaining is less than another system defined value (low water mark), the system transmits a START character which is intended to cause the terminal device to resume transmitting data (without risk of overflowing the input queue). In order to avoid potential deadlock, **IXOFF** is ignored in canonical mode whenever there is no line delimiter in the input buffer. In this case, the STOP character is not sent at the high water mark, but will be transmitted later if a delimiter is received. If all complete lines are read from the input queue leaving only a partial line with no line delimiter, the START character is sent, even if the number of characters is still greater than the low water mark. When **ICANON** is set and the input stream contains more characters between line delimiters than the high water mark allows, there is no guarantee that **IXOFF** can prevent buffer overflow and data loss, because the STOP character may not be sent in time, if at all.

If **IMAXBEL** is set, the ASCII BEL character is echoed if the input queue overflows. Further input is not stored, but any input present in the input queue is not discarded. If **IMAXBEL** is clear, no ASCII BEL character is echoed, and the input already present in the input queue is discarded when the input queue overflows.

The initial input control value is all bits clear.

Output Modes

The **c_oflag** field specifies the system treatment of output:

OPOST Postprocess output.

OLCUC	Map lowercase to uppercase on output.
ONLCR	Map NL to CR-NL on output.
OCRNL	Map CR to NL on output.
ONOCR	No CR output at column 0.
ONLRET	NL performs CR function.
OFILL	Use fill characters for delay.
OFDEL	Fill is DEL, else NUL.
NLDLY	Select new-line delays:
NL0	No delay
NL1	Delay type 1
CRDLY	Select carriage-return delays:
CR0	No delay
CR1	Delay type 1
CR2	Delay type 2
CR3	Delay type 3
TABDLY	Select horizontal-tab delays:
TAB0	No delay
TAB1	Delay type 1
TAB2	Delay type 2
TAB3	Expand tabs to spaces.
XTABS	Expand tabs to spaces.
BSDLY	Select backspace delays:
BS0	No delay
BS1	Delay type 1
VTDLY	Select vertical-tab delays:
VT0	No delay
VT1	Delay type 1
FFDLY	Select form-feed delays:
FF0	No delay
FF1	Delay type 1

If **OPOST** is set, output characters are post-processed as indicated by the remaining flags; otherwise characters are transmitted without change.

If **OLCUC** is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with **IUCLC**.

If **ONLCR** is set, the NL character is transmitted as the CR-NL character pair. If **OCRNL** is set, the CR character is transmitted as the NL character. If **ONOCR** is set, no CR character is transmitted when at column 0 (first position). If **ONLRET** is set, the NL character is assumed to do the carriage-return function; the column pointer will be set to 0, and the delays specified for CR will be used. If **ONLRET** is clear, the NL character is assumed to perform only the line-feed function; the delays specified for NL are used and the column pointer remains unchanged. For all of these cases, the column pointer is always set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. The values of **NL0**, **CR0**, **TAB0**, **BS0**, **VT0**, and **FF0** indicate no delay. If **OFILL** is set, fill characters are transmitted for delay instead of a timed delay. This is useful for high baud rate terminals, that need only a minimal delay. If **OFDEL** is set, the fill character is DEL; otherwise NUL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If **ONLRET** is set, carriage-return delays are used instead of the new-line delays. If **OFILL** is set, two fill characters are transmitted.

Carriage-return delay type 1 depends on the current column position; type 2 is about 0.10 seconds; type 3 about 0.15 seconds. If **OFILL** is set, delay type 1 transmits two fill characters; type 2, four fill characters.

Horizontal-tab delay type 1 is depends on the current column position. Type 2 is about 0.10 seconds; type 3 specifies that tabs are to be expanded into spaces. If **OFILL** is set, two fill characters are transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If **OFILL** is set, one fill character is transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

Control Modes

The `c_cflag` field describes the hardware control of the terminal:

CBAUD	Baud rate:	CSIZE	Character size:
B0	Hang up	CS5	5 bits
B50	50 baud	CS6	6 bits
B75	75 baud	CS7	7 bits
B110	110 baud	CS8	8 bits
B134	134.5 baud		
B150	150 baud	CSTOPB	Send two stop bits, else one.
B200	200 baud	CREAD	Enable receiver.
B300	300 baud	PARENB	Parity enable.
B600	600 baud	PARODD	Odd parity, else even.
B900	900 baud	HUPCL	Hang up on last close.
B1200	1200 baud	CLOCAL	Local line, else dial-up.
B1800	1800 baud	LOBLK	Reserved for use by <i>shl(1)</i> .
B2400	2400 baud		
B3600	3600 baud		
B4800	4800 baud		
B7200	7200 baud		
B9600	9600 baud		
B19200	19200 baud		
B38400	38400 baud		
EXTA	External A		
EXTB	External B		

The **CBAUD** bits specify the baud rate. The zero baud rate, **B0**, is used to hang up the connection. If **B0** is specified, the modem control lines (see *modem(7)*) cease to be asserted. Normally, this disconnects the line. For any particular hardware, impossible speed changes are ignored. **CBAUD** is provided for use with the **termio** structure. When the **termios** structure is used, several routines are available for setting and getting the input and output baud rates (see *termios Structure Related Functions*).

The **CSIZE** bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If **CSTOPB** is set, two stop bits are used; otherwise one stop bit. For example, at 110 baud, many devices require two stop bits.

If **PARENB** is set, parity generation is enabled (a parity bit is added to each output character). Furthermore, parity detection is enabled (incoming characters are checked for the correct parity). If **PARENB** is set, **PARODD** specifies odd parity if set; otherwise even parity is used. If **PARENB** is clear, both parity generation and parity checking are disabled.

If **CREAD** is set, the receiver is enabled. Otherwise no characters can be received.

The specific effects of the **HUPCL** and **CLOCAL** bits depend on the mode and type of the modem control in effect. See *modem(7)* for the details.

If **HUPCL** is set, the modem control lines for the port are lowered (disconnected) when the last process using the open port closes it or terminates.

If **CLOCAL** is set, a connection does not depend on the state of the modem status lines. If **CLOCAL** is clear, the modem status lines are monitored.

Under normal circumstances, a call to `read()` waits for a modem connection to complete. However, if either the **O_NDELAY** or the **O_NONBLOCK** flags are set or **CLOCAL** is set, the `open()` returns immediately without waiting for the connection. If **CLOCAL** is set, see *Modem Disconnect* for the effects of `read()` and `write()` for those files for which the connection has not been established or has been lost.

LOBLK is used by the shell layers facility (see *shl(1)*). The shell layers facility is not part of the general terminal interface, and the **LOBLK** bit is not examined by the general terminal interface.

The initial hardware control value after open is **B300**, **CS8**, **CREAD**, and **HUPCL**.

Local Modes

The `c_lflag` field is used to control terminal functions.

ISIG	Enable signals.
ICANON	Canonical input (erase and kill processing).
XCASE	Canonical upper/lower presentation.
ECHO	Enable echo.
ECHOE	Echo ERASE as correcting backspace sequence.
ECHOK	Echo NL after kill character.
ECHONL	Echo NL.
NOFLSH	Disable flush after interrupt, quit, or suspend.
TOSTOP	Send SIGTTOU for background output.
ECHOCTL	Echo control characters as ^char, DEL as ^?.
ECHOPRT	Echo erased character as character is erased.
ECHOKE	BS SP BS erase entire line on line kill.
FLUSHO	Output is being flushed.
PENDIN	Reprocess pending input at next read or input character.
TEXTEN	Enable extended functions.

If **ISIG** is set, each input character is checked against the special control characters INTR, QUIT, SUSP, and DSUSP (see *Process Group Control IOCTL Commands*). If an input character matches one of these control characters, the function associated with that character is performed and the character is discarded. If **ISIG** is clear, no checking is done and the character is treated as a normal data character. Thus these special input functions are possible only if **ISIG** is set.

If **ICANON** is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, EOL, or EOL2. If **ICANON** is clear, read requests are satisfied directly from the input queue. A read blocks until at least MIN characters have been received or the timeout value TIME has expired between characters. (See *Non-Canonical Mode Input Processing (MIN/TIME Interaction)*). This allows fast bursts of input to be read efficiently while still allowing single-character input. The time value represents tenths of seconds.

If **XCASE** is set, and if **ICANON** is set, an uppercase letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

To obtain:	Use:
\	\\
	\\
{	\\{
}	\\}
\	\\

For example, **A** is input as \a, \n as \\n, and \N as \\N. **XCASE** would normally be used in conjunction with **IUCLC** and **OLCUC** for terminals that support only the first-sixty-four-character limited character set. In this case, **IUCLC** processing is done before **XCASE** for input, and processing is done after **XCASE** for output. Therefore typing **A** causes an **a** to be read because of **IUCLC**, and typing \A causes an **A** to be read since **IUCLC** produces \a which is turned into **A** by the **XCASE** processing.

If **ECHO** is set, characters are echoed back to the terminal when received. If **ECHO** is clear, characters are not echoed.

When **ICANON** is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, EOL and EOL2 as described in *Canonical Mode Input Processing*. Furthermore, the following echo functions are possible.

If **ECHO** and **ECHOE** are set, the ERASE and WERASE characters are echoed as the three-character ASCII sequence BS SP BS, which clears the last character or word from the CRT screen.

If **ECHO** and **ECHOPRT** are set, and **ECHOE** is clear, the first ERASE and WERASE character in a sequence echoes a backslash (\) followed by the characters being erased. Subsequent ERASE or WERASE characters echo the characters being erased in reverse order. The next non-erase character causes a slash (/) to be typed before it is echoed.

If **ECHOKE** and **ECHO** are set, the KILL character is echoed by erasing each character on the line from the CRT screen using the method selected by **ECHOE** and **ECHOPRT**.

If **ECHOCTL** and **ECHO** are set, all control characters (characters with codes between 0 and 37 octal) other than ASCII TAB, ASCII NL, the START and STOP characters, ASCII CR, and ASCII BS are echoed as ^char, where char is the character given by adding 100 octal to the control character's code.

If **ECHOK** is set and **ECHOKE** is not set, the NL character is echoed after the kill character to emphasize that the line is being deleted.

If **ECHONL** is set, the NL character is echoed even if **ECHO** is clear. This is useful for terminals set to local echo (that is, half duplex).

Unless escaped, the EOF character is not echoed. Because ASCII EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If **NOFLSH** is set, the normal flush of the input and output queues associated with quit, interrupt, and suspend characters is not done. However, **NOFLSH** does not affect the flushing of data upon receipt of a break when **BRKINT** is set.

If the **TOSTOP** bit is set, an attempt by a process that is not in the foreground process group to write to its controlling terminal will be denied when the process is not ignoring and not blocking the **SIGTTOU** signal. If the write is denied and the process is a member of an orphaned process group **write()** returns **-1** and sets **errno** to **EIO** and no signal is sent. If the write is denied and the process is a not a member of an orphaned process group, the **SIGTTOU** signal is sent to that process group.

If **FLUSHO** is set, data written to the terminal device is discarded. This bit is set by a program. A program can cancel the **FLUSHO** effect by clearing **FLUSHO**.

If **PENDIN** is set, any input that has not been read is reprocessed and possibly re-echoed when the next character arrives as input.

If **ICANON** is set, the ERASE, KILL, and EOF characters can be escaped by a preceding **** character, in which case no special function is done.

IEXTEN must be set before the **ECHOCTL**, **ECHOPRT**, **ECHOKE**, **FLUSHO**, and **PENDIN** functions are allowed. In addition, the special characters **WERASE** and **LNEXT** are allowed only if **IEXTEN** is set. **IEXTEN** does not affect any other functions.

The initial local control value is all-bits-clear.

Special Control Characters

Special control characters are defined in the array **c_cc**. All of these special characters can be changed. The subscript name and description for each element in both canonical and non-canonical mode are shown in the following table.

Canonical	Subscript Usage Non-Canonical	Description
VEOF		EOF character
VEOL		EOL character
VEOL2		EOL2 character
VERASE		ERASE character
VWERASE		WERASE character
VINTR	VINTR	INTR character
VKILL		KILL character
	VMIN	MIN value
VQUIT	VQUIT	QUIT character
VSTART	VSTART	START character
VSTOP	VSTOP	STOP character
VSUSP	VSUSP	SUSP character
VDSUSP	VDSUSP	DSUSP character
	VTIME	TIME value
VLNEXT	VLNEXT	LNEXT character

termios Structure-Related Functions

The following functions are provided when using the *termios* structure. Note that the effects on the terminal device of the **cfsetispeed()** and **cfsetospeed()** functions do not become effective until the **tcsetattr()** function is successfully called. Refer to the appropriate manual entries for details.

termios Structure Functions	
Function	Description
cfgetospeed()	get output baud rate
cfgetispeed()	get input baud rate
cfsetospeed()	set output baud rate

cfsetispeed()	set input baud rate
tcgetattr()	get terminal state
tcsetattr()	set terminal state

termio Structure-Related IOCTL Commands

Several `ioctl()` system calls apply to terminal files that use the `termio` structure (see *termio Structure*). If a requested command is not recognized, the request returns `-1` with `errno` set to `EINVAL`.

`ioctl()` system calls that reference the `termio` structure have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

Commands using this form are:

TCGETA	Get the parameters associated with the terminal and store them in the <code>termio</code> structure referenced by <i>arg</i> . This command is allowed from a background process; however, the information may be subsequently changed by a foreground process.
TCSETA	Set the parameters associated with the terminal from the <code>termio</code> structure referenced by <i>arg</i> . The change is immediate. If characters are being output when the command is requested, results are undefined and the output may be garbled.
TCSETAW	Wait for the output to drain before setting new parameters. This form should be used when changing parameters that affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

termio Caveats

Only the first eight special control characters (see *termios Structure*) can be set or returned. The values of indices `VEOL` and `VEOF` are the same as indices `VTIME` and `VMIN` respectively. Hence if `ICANON` is set, `VEOL` or `VTIME` is the additional end-of-line character and `VEOF` or `VMIN` is the end-of-file character. If `ICANON` is clear, `VEOL` or `VTIME` is the inter-character-timer value and `VEOF` or `VMIN` is the minimum number of characters desired for reads.

Structure-Independent Functions

The following functions which are independent of both the `termio` and `termios` structures are provided for controlling terminals. Refer to the appropriate manual entries for details.

Function	Description
<code>tcsendbreak()</code>	send a break
<code>tcdrain()</code>	wait until output has drained
<code>tcflush()</code>	flush input or output queue or both
<code>tcflow()</code>	suspend or resume input or output
<code>tcgetpgrp()</code>	get foreground process group id
<code>tcsetpgrp()</code>	set foreground process group id
<code>tcgetsid()</code>	get session id

System Asynchronous I/O IOCTL Commands

The following `ioctl()` system calls provide for system asynchronous I/O and have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIOSSAIOSTAT	If the integer referenced by <i>arg</i> is non-zero, system asynchronous I/O is enabled; that is, enable <code>SIGIO</code> to be sent to the process currently designated with <code>FIOSSAIOOWN</code> (see below) whenever the terminal device file status changes from "no read data available" to "read data available". If no process has been designated with <code>FIOSSAIOOWN</code> , enable <code>SIGIO</code> to be sent to the first process that opened the terminal device file.
---------------------	---

If the designated process has exited, the `SIGIO` signal is not sent to any process.

If the integer referenced by *arg* is 0, system asynchronous I/O is disabled.

	The default on open of a terminal device file is that system asynchronous I/O is disabled.
FIOGSAIOSTAT	The integer referenced by <i>arg</i> is set to 1 if system asynchronous I/O is enabled. Otherwise, the integer referenced by <i>arg</i> is set to 0.
FIOSSAIOOWN	Set the process ID that will receive the SIGIO signals due to system asynchronous I/O to the value of the integer referenced by <i>arg</i> . If no process can be found corresponding to that specified by the integer referenced by <i>arg</i> , the call returns -1 with errno set to ESRCH . A user with appropriate privileges can designate that any process receive the SIGIO signals. If the request is not made by a user with appropriate privileges and the calling process does not either designate that itself or another process whose real, saved, or effective user ID matches its real or effective user ID or the calling process does not designate a process that is a descendant of the calling process to receive the SIGIO signals, the call returns -1 with errno set to EPERM . If the designated process subsequently exits, the SIGIO signal is not sent to any process. The default on open of a terminal device file is that the process performing the first open is set to receive the SIGIO signals.
FIOGSAIOOWN	The integer referenced by <i>arg</i> is set to the process ID designated to receive SIGIO signals.

Line Control IOCTL Commands

Several `ioctl()` system calls control input and output. Some of these calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

Commands using this form are:

TCSBRK	Wait for the output to drain. If <i>arg</i> is 0, send a break (zero bits for at least 0.25 seconds). The <code>tcsendbreak()</code> function performs the same function (see <code>tcsendbreak(3C)</code>).
TCXONC	Start/stop control. If <i>arg</i> is 0, suspend output; if 1, restart suspended output; if 2, transmit a STOP character; if 3, transmit a START character. If any other value is given for <i>arg</i> , the call returns -1 with errno set to EINVAL . The <code>tcflow()</code> function performs the same functions (see <code>tcflow(3C)</code>).
TCFLSH	If <i>arg</i> is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues. If any other value is given for <i>arg</i> , the call returns -1 with errno set to EINVAL . The <code>tcflush()</code> function performs the same functions (see <code>tcflush(3C)</code>).

Sending a **BREAK** is accomplished by holding the data transmit line at a SPACE or logical zero condition for at least 0.25 seconds. During this interval, data can be sent to the device, but because of serial data interface limitations, the **BREAK** takes precedence over all data. Thus, all data sent to a device during a **BREAK** is lost. This includes system-generated **XON/XOFF** characters used for input flow control. Note also that a delay in transmission of the **XOFF** flow control character until after the **BREAK** is terminated could still result in data overflow because the flow control character may not be sent soon enough.

Other calls have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIONREAD	Returns in the integer referenced by <i>arg</i> the number of characters immediately readable from the terminal device file. This command is allowed from a background process; however, the data itself cannot be read from a background process.
-----------------	--

Non-blocking I/O IOCTL Commands

Non-blocking I/O is easily provided via the `O_NONBLOCK` and `O_NDELAY` flags available in both `open(2)` and `fcntl(2)`. The commands in this section are provided for backward compatibility with previously developed applications. `ioctl()` system calls that provide a style of non-blocking I/O different from `O_NONBLOCK` and `O_NDELAY` have the form:

```
ioctl (fildes, command, arg)
int *arg;
```

Commands using this form are:

FIOENBIO If the integer referenced by *arg* is non-zero, **FIOENBIO**-style non-blocking I/O is enabled; that is, subsequent reads and writes to the terminal device file are handled in a non-blocking manner (see below). If the integer referenced by *arg* is 0, **FIOENBIO**-style non-blocking I/O is disabled.

For reads, **FIOENBIO**-style non-blocking I/O prevents all read requests to that device file from blocking, whether the requests succeed or fail. Such a read request completes in one of three ways:

- If there is enough data available to satisfy the entire request, the read completes successfully, having read all of the data, and returns the number of characters read;
- If there is not enough data available to satisfy the entire request, the read completes successfully, having read as much data as possible, and returns the number of characters read;
- If there is no data available, the read returns `-1` with `errno` set to `EWOLDBLOCK`.

For writes, **FIOENBIO**-style non-blocking I/O prevents all write requests to that device file from blocking, whether the requests succeed or fail. Such a write request completes in one of three ways:

- If there is enough space available in the system to buffer all the data, the write completes successfully, having written out all of the data, and returns the number of characters written;
- If there is not enough space in the buffer to write out the entire request, the write completes successfully, having written as much data as possible, and returns the number of characters written;
- If there is no space in the buffer, the write returns `-1` with `errno` set to `EWOLDBLOCK`.

To prohibit **FIOENBIO**-style non-blocking I/O from interfering with the `O_NONBLOCK` and `O_NDELAY` flags (see `open(2)` and `fcntl(2)`), the functionality of `O_NONBLOCK` and `O_NDELAY` always supersedes the functionality of **FIOENBIO**-style non-blocking I/O. This means that if either `O_NONBLOCK` or `O_NDELAY` is set, the driver performs read requests in accordance with the definition of `O_NDELAY` or `O_NONBLOCK`. When both `O_NONBLOCK` and `O_NDELAY` are clear, the definition of **FIOENBIO**-style non-blocking I/O applies.

The default on open of a terminal device file is that **FIOENBIO**-style non-blocking I/O is disabled.

FIOGSNBIO The integer referenced by *arg* is set to 1, if **FIOENBIO**-style non-blocking I/O is enabled. Otherwise, the integer referenced by *arg* is set to 0.

Process Group Control IOCTL Commands

The process group control features described here (except for setting and getting the delayed stop process character) are easily implemented using the functions `tcgetattr()`, `tcsetattr()`, `tcgetpgrp()`, `tcsetpgrp()`, and `tcsetsid()`, (see `tcattribute(3C)`, `tcgetpgrp(3C)`, `tcsetpgrp(3C)`, and `tcsetsid(3C)` respectively).

The following structure, used with process group control, is defined in `<bsd/tty.h>`:

```
struct ltchars {
    unsigned char t_suspc;    /* stop process character*/
```

```

    unsigned char t_dsuspc; /* delayed stop process character*/
    unsigned char t_rprntc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_flushc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_werasc; /* reserved; must be '_POSIX_VDISABLE'*/
    unsigned char t_lnextc; /* reserved; must be '_POSIX_VDISABLE'*/
};

```

The initial value for all these characters is `_POSIX_VDISABLE`, which causes them to be disabled. The meaning for each character is as follows:

- t_suspc** Suspend the foreground process group. A *suspend* signal (`SIGTSTP`) is sent to all processes in the foreground process group. Normally, each process is forced to stop, but arrangements can be made to either ignore or block the signal, or to receive a trap to an agreed-upon location; see *signal(2)* and *signal(5)*. When enabled, the typical value for this character is Control-Z or ASCII SUB. Setting or getting `t_suspc` is equivalent to setting or getting the SUSP special control character.
- t_dsuspc** Same as `t_suspc`, except that the *suspend* signal (`SIGTSTP`) is sent when a process reads the character, rather than when the character is typed. When enabled, the typical value for this character is Ctrl-Y or ASCII EM.

Attempts to set any of the reserved characters to a value other than `_POSIX_VDISABLE` cause `ioctl()` to return `-1` with `errno` set to `EINVAL` with no change in value of the reserved character.

`ioctl()` system calls that use the above structure have the form:

```

ioctl (fildes, command, arg)
struct ltchars *arg;

```

Commands using this form are:

- TIOCG LTC** Get the process group control characters and store them in the *ltchars* structure referenced by *arg*. This command is allowed from a background process. However, the information may be subsequently changed by a foreground process.
- TIOCS LTC** Set the process group control characters from the structure referenced by *arg*.

Additional process group control `ioctl()` system calls have the form:

```

ioctl (fildes, command, arg)
unsigned int *arg;

```

Commands using this form are:

- TIOCGPGRP** Returns in the integer referenced by *arg* the foreground process group associated with the terminal. This command is allowed from a background process. However, the information may be subsequently changed by a foreground process. This feature is easily implemented using the `tcgetpgrp()` function (see *tcgetpgrp(3C)*).

If the `ioctl()` call fails, it returns `-1` and sets `errno` to one of the following values:

- [EBADF] *fildes* is not a valid file descriptor.
- [ENOTTY] The file associated with *fildes* is not the controlling terminal, or the calling process does not have a controlling terminal.
- [EACCES] The file associated with *fildes* is the controlling terminal of the calling process, however, there is no foreground process group defined for the controlling terminal.

Note: EACCES may not be returned in future releases. Behavior in cases where no foreground process group is defined for the controlling terminal may change in future versions of the POSIX standard. Portable applications, therefore, should not rely on this error condition.

- TIOCS PGRP** Sets the foreground process group associated with the terminal to the value referenced by *arg*. This feature is easily implemented using the `tcsetpgrp()` function (see *tcsetpgrp(3C)*).

If the `ioctl()` call fails, it returns `-1` and sets `errno` to one of the following values:

[EBADF]	<i>fildev</i> is not a valid file descriptor.
[EINVAL]	The process ID referenced by <i>arg</i> is not a supported value.
[ENOTTY]	The calling process does not have a controlling terminal, or the <i>fildev</i> is not the controlling terminal, or the controlling terminal is no longer associated with the session of the calling process.
[EPERM]	The process ID referenced by <i>arg</i> is a supported value but does not match the process group ID of a process in the same session as the calling process.

TIOCGSID Returns in the integer referenced by *arg* the session ID of the terminal specified by *fildev*. This feature is easily implemented using the `tcgetsid()` function (see `tcgetsid(3C)`).

If the `ioctl()` call fails, it returns `-1` and sets `errno` to one of the following values:

[EBADF]	<i>fildev</i> is not a valid file descriptor.
[ENOTTY]	The device associated with <i>fildev</i> is not a terminal.
[EACCES]	The <i>fildev</i> is a terminal that is not allocated to a session.

TIOCLGET Get the process group control mode word and store it in the int referenced by *arg*. This command is allowed from a background process; however, the information may be subsequently changed by a foreground process.

TIOCLSET Set the process group control mode word to the value of the int referenced by *arg*.

TIOCLBIS Use the int referenced by *arg* as a mask of bits to set in the process group control mode word.

TIOCLBIC Use the int referenced by *arg* as a mask of bits to clear in the process group control mode word.

The following bit is defined in the process group control mode word:

LTOSTOP Send `SIGTTOU` for background writes.

Setting or clearing `LTOSTOP` is equivalent to setting or clearing the `TOSTOP` flag (see *Local Modes*). If `LTOSTOP` is set and a process is not in the foreground process group of its controlling terminal, a write by the process to its controlling terminal may be denied (see *Terminal Access Control*).

Terminal Size IOCTL Commands

The following `ioctl()` system calls are used to get and set terminal size information for the terminal referenced by *fildev*. These `ioctl()` system calls use the `winsize` structure to get and set the terminal size information. The `winsize` structure, defined in `<termios.h>`, has the following members :

```
unsigned short ws_row;      /* Rows, in characters */
unsigned short ws_col;      /* Columns, in characters */
unsigned short ws_xpixel;   /* Horizontal size, in pixels */
unsigned short ws_ypixel;   /* Vertical size, in pixels */
```

The initial values for all elements of terminal size are zero. The values for terminal size are neither set nor used by the general terminal interface, and have no effect on the functionality of the general terminal interface. The values for terminal size are set and used only by applications that access them through the terminal-size `ioctl()` system calls (see `ioctl(2)`).

`ioctl()` system calls that use the above structure have the form:

```
ioctl (fildev, command, arg)
struct winsize *arg;
```

Commands using this form are:

TIOCGWINSZ Get the terminal size values and store them in the `winsize` structure referenced by *arg*. This command is allowed from a background process.

TIOCSWINSZ Set the terminal size values from the **winsize** structure referenced by *arg*. If any of the new values differ from previous values, a **SIGWINCH** signal is sent to all processes in the terminal's foreground process group.

Console Output Redirection IOCTL Command

Output which would normally be sent to the system console may be redirected to any other TTY device or pseudo-device in the system. The **ioctl()** system call used to control console output redirection has the form:

```
ioctl (fildes, command, arg)
int arg;
```

The command using this form is:

TIOCCONS Redirect system console output. Any output that would normally be sent to the system console, either through kernel **printf** requests, or through the console special file, will instead be sent to the terminal referenced by *fildes*. The value of *arg* is ignored. The user must have superuser privileges to execute this request. Otherwise, the call returns **-1** with **errno** set to **EPERM**. If the console output has not been redirected to a different device by a later call to this command, it is redirected back to the physical console device when *fildes* is closed.

WARNINGS

Various HP-UX implementations use non-serial interfaces that look like terminals (such as bit-mapped graphics displays) or "smart cards" that cannot implement the exact capabilities described above. Therefore, not all systems can exactly meet the standard stated above. Each implementation is required to state any deviations from the standard as part of its system-specific documentation.

FIOSSAIOSTAT	is similar to BSD 4.2 FIOASYNC , with the addition of provisions for security.
FIOGSAIOSTAT	is of HP origin, complements FIOSSAIOSTAT , and allows saving and restoring system asynchronous I/O TTY states for command interpreter processes.
FIOSSAIOOWN	is similar to BSD 4.2 FIOSETOWN , with additional provisions for security.
FIOGSAIOOWN	is similar to BSD FIOGETOWN . 4.2 Note also the difference that the BSD 4.2 version of this functionality used process groups, while the HP-UX version only uses processes.
FIOSNBIO	is the same as BSD FIONBIO , 4.2 except that it does not interfere with the O_NDELAY or O_NONBLOCK open() and fcntl() flags.
FIOGNBIO	is of HP origin, complements FIOSNBIO , and allows saving and restoring the FIOSNBIO -style non-blocking I/O TTY state for command interpreter processes.

The general terminal interface uses a system resource known as a **cblock** to store data being transmitted or received through a communications port. These cblocks are continuously used and freed for reuse as data pass through the system. If too few cblocks are configured in the system, the cblock pool may be temporarily or permanently exhausted, and data loss, system hangs, or reduced system performance can result.

If cblock exhaustion is suspected, you can examine the system message buffer with **dmesg** (see **dmesg(1M)**) for messages indicating cblock exhaustion has occurred. Or, you can use **adb** (see **adb(1)**) if examining the corefile of a dump. The message format is

WARNING: cblock exhaustion occurred n times

where *n* indicates the number of times the operating system has requested a cblock and none could be provided. If this message is observed, the kernel should be reconfigured to generate a larger number of cblocks.

A cblock is 32 bytes in length. The default number of cblocks configured in the system is computed from the tunable system parameter **maxusers**, and is defined to be

100+16*maxusers

This can be overridden by using the optional tunable system parameter **nclist** to specify the desired number of cblocks to be used in the system.

SAM or **config(1M)** may be used to generate a new kernel with a new **nclist** value.

DEPENDENCIES

Series 700

Built-in serial ports on Series 700 machines support the following additional baud rate settings: 57 600, and 115 200. An RS-232-to-RS-422 converter may be required to achieve practical cable lengths at these baud rates (because RS-232 only specifies up to 19 200 baud).

Timed delays are not supported.

Built-in serial ports on Series 700 systems have RTS and CTS flow control capability, configurable receive FIFO trigger levels, and a configurable transmit limit. RTS/CTS hardware handshaking can be enabled through a bit in the device file minor number, through an `ioctl()` call (see `termiox(7)`), or through the `stty` command (see `stty(1)`).

The receive FIFO trigger level is configurable through two bits in the device file minor number. The receive FIFO trigger level is used to set the level at which a receive interrupt is generated to the system. Setting a smaller value for the receive FIFO trigger level enables the system to react more quickly to receipt of characters. However, using a smaller trigger level increases system overhead to process the additional interrupts. A higher receive FIFO trigger level reduces the system interrupt overhead for heavy inbound data traffic at the cost of less time for the system to read data from the hardware before receive FIFOs are overrun. When using RTS flow control, the receive FIFO trigger level also determines the point at which the hardware lowers RTS to protect the receive FIFO. Use of a higher receive FIFO trigger level also reduces XOFF flow control responsiveness because, under light inbound data flow conditions, receipt of the XOFF character by the system is slightly delayed. Choice of the appropriate receive FIFO trigger level should be based upon how the serial port is to be used. For most applications a receive FIFO trigger level of 8 (`c3,c2 = 10`) is suggested.

Two bits in the device file minor number specify the transmit limit, the number of characters which are successively loaded into the transmit FIFO. Setting a smaller transmit limit allows the transmitter to be more responsive to flow control either from receipt of an XOFF character or de-assertion of CTS at the cost of increased system interrupt overhead. Setting a larger transmit limit reduces interrupt overhead but is not as responsive to flow control since the remainder of the transmit FIFO can be transmitted even after the transmitter is flow controlled. When communicating with devices which have little tolerance for data receipt after flow control, one must choose the transmit limit appropriately.

Device file minor number

Series 700 device file minor numbers take the form:

`0xIIC0HM`

where:

- `II` = Two hexadecimal digits (8 bits) to indicate the instance of the serial interface.
- `C` = One hexadecimal digit (4 bits) for FIFO control. Values for each bit are as follows:

Receive FIFO Trigger Level			Transmit Limit		
<code>c₃</code>	<code>c₂</code>	Level	<code>c₁</code>	<code>c₀</code>	Limit
0	0	1	0	0	1
0	1	4	0	1	4
1	0	8	1	0	8
1	1	14	1	1	12

- `H` = One hexadecimal digit (4 bits) which controls diagnostic access and hardware flow control.

Bit	Value
<code>h₃</code>	Diagnostic telephony access
<code>h₂</code>	Reserved
<code>h₁</code>	Reserved
<code>h₀</code>	Enables RTS/CTS hardware flow control

- `M` = One hexadecimal digit (4 bits) to determine the port access type. Values for each bit are as follows:

Bit	Value
m ₃	TI/ALP
m ₂	0 = Simple protocol (U.S.), 1 = CCITT protocol (Europe)
m ₁ m ₀	00 = Direct 01 = Dial-out modem 10 = Dial-in modem 11 = Invalid

Series 800

Timed output delays are not directly supported. If used, an appropriate number of fill characters (based on the current baud rate) is output. The total time to output the fill characters is at least as long as the time requested.

The system specified input flow control values are as follows: low water mark is 60, high water mark is 180, and maximum allowed input is 512.

The HP 98196A (formerly 27140A option 800) interface does not support the following hardware settings:

CBAUD B200, B38400, EXTA, EXTB.

The HP A1703-60003 and the HP 28639-60001 interfaces do not support baud rates above 9600. Furthermore, changing the following hardware settings on port 0 from the default (9600 baud, 8 bit characters, 1 stop bit, no parity) is not supported:

CBAUD, CSIZE, CSTOPB, PARENB, PARODD.

The HP J2094A interface does not support baud rates above 19200.

The HP J2094A supports RTS and CTS flow control. The RTS/CTS hardware handshaking can be enabled through a bit in the device file minor number, through an `ioctl()` call (see *termiox(7)*), or through the `stty` command (see *stty(1)*).

Device file minor number

Series 800 device file minor numbers take the form:

0xIIPPHM

where:

- II** = Two hexadecimal digits (8 bits) to indicate the instance of the serial interface.
- PP** = Two hexadecimal digits (8 bits) to indicate the port number of this device on the serial interface.
- H** = One hexadecimal digit (4 bits) which controls diagnostic access and hardware flow control (HP J2094A only).

Bit	Value
h ₃	Card diagnostic
h ₂	Port diagnostic
h ₁	Reserved
h ₀	Enables RTS/CTS hardware flow control

M = One hexadecimal digit (4 bits) for the port access type. Values for each bit are as follows:

Bit	Value
m ₃	TI/ALP
m ₂	0 = Simple protocol (U.S.), 1 = CCITT protocol (Europe)
m ₁ m ₀	00 = Direct 01 = Dial-out modem 10 = Dial-in modem 11 = Invalid

AUTHOR

termios was developed HP and the IEEE Computer Society.

termio was developed by HP, AT&T, and the University of California, Berkeley.

FILES

```
/dev/console
/dev/cua*
/dev/cul*
/dev/tty*
/dev/ttyd*
```

EISA mux has adopted the following device files naming convention:

```
/dev/<cua|cul|tty|ttyd><Instance_Number><Port_Module_Name><Port_Number>
```

where *Port_Module_Name* = [a..h], *Port_Number* = 1-16

For example: `/dev/cua2a1`, `/dev/cul2b8`, `/dev/tty2g12`, `/dev/ttyd2h16`

SEE ALSO

adb(1), shl(1), stty(1), config(1M), dmesg(1M), mknod(1M), fork(2), ioctl(2), setsid(2), signal(2), stty(2), setpgid(2), blmode(3C), cfspeed(3C), tccontrol(3C), tcattribute(3C), tcgetpgrp(3C), tcsetpgrp(3C), tcgetsid(3C), signal(5), unistd(5), sttyV6(7), tty(7), modem(7), termiox(7).

STANDARDS CONFORMANCE

termio: SVID2, SVID3, XPG2

termios: AES, SVID3, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

termiox - extended general terminal interface

SYNOPSIS

```
#include <sys/termiox.h>
```

```
ioctl (int fildes, int request, struct termiox * arg)
```

DESCRIPTION

The extended general terminal interface supplements the *termio(7)* general terminal interface by adding support for asynchronous hardware flow control and local implementations of additional asynchronous features. Some systems may not support all of these capabilities because of hardware or software limitations. Other systems may not permit certain functions to be disabled. In such cases, the appropriate bits are ignored. If the capabilities can be supported, the interface described here must be used.

Hardware Flow Control Modes

Hardware flow control supplements the *termio IXON*, *IXOFF*, and *IXANY* character flow control (see *termio(7)*). Character flow control occurs when one device controls the data transfer of another device by inserting control characters in the data stream between devices. Hardware flow control occurs when one device controls the data transfer of another device by using electrical control signals on wires (circuits) of the asynchronous interface. Character flow control and hardware flow control can be simultaneously set.

In asynchronous, full duplex applications, the use of the Electronics Industries Association's EIA-232-D Request To Send (RTS) and Clear To Send (CTS) circuits is the preferred method of hardware flow control.

The EIA-232-D standard specified only unidirectional hardware flow control where the Data Circuit-terminating Equipment or Data Communications Equipment (DCE) indicates to the Data Terminal Equipment (DTE) to stop transmitting data. The *termiox* interface allows both unidirectional and bidirectional hardware flow control; when bidirectional flow control is enabled, either the DCE or DTE can indicate to each other to stop transmitting data across the interface.

Clock Modes

Isochronous flow control and clock mode communication are not supported.

Terminal Parameters

Parameters that control the behavior of devices providing the *termiox* interface are specified by the *termiox* structure, defined in the *<sys/termiox.h>* header file. Several *ioctl()* system calls (see *ioctl(5)*) that fetch or change these parameters use the *termiox* structure which contains the following members:

```
unsigned short x_hflag;      /* hardware flow control modes */
unsigned short x_cflag;      /* clock modes */
unsigned short x_rflag;      /* reserved modes */
unsigned short x_sflag;      /* spare local modes */
```

The *x_hflag* field describes hardware flow control modes:

RTSXOFF	0000001	Enable RTS hardware flow control on input.
CTSXON	0000002	Enable CTS hardware flow control on input.

The RTS and CTS circuits are involved in establishing CCITT modem connections. Since RTS and CTS circuits are used both by CCITT modem connections and by hardware flow control, CCITT modem and hardware flow control cannot be simultaneously enabled.

Variations of different hardware flow control methods can be selected by setting the appropriate bits. For example, bidirectional RTS/CTS flow control is selected by setting both the RTSXOFF and CTSXON bits. Unidirectional CTS hardware flow control is selected by setting only the CTSXON bit.

If RTSXOFF is set, the Request to Send (RTS) circuit (line) is raised, and if the asynchronous port needs to have its input stopped, it lowers the Request to Send (RTS) line. If the RTS line is lowered, it is assumed that the connected device will stop its output until RTS is raised.

If CTSXON is set, output occurs only if the Clear To Send (CTS) circuit (line) is raised by the connected device. If the CTS line is lowered by the connected device, output is suspended until CTS is raised.

termiox Structure Related IOCTL Command

The `ioctl()` system calls that reference the `termiox` structure have the form:

```
ioctl (fildes, command, arg)
struct termiox *arg;
```

Commands using this form are:

TCGETX	The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are fetched and stored into that structure.
TCSETX	The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change is immediate. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices.
TCSETXW	The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that affect output. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices.
TCSETXF	The argument is a pointer to a <code>termiox</code> structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded, then the change occurs. Errors that can be returned include: <ul style="list-style-type: none"> [EINVAL] The port does not support hardware flow control. [ENOTTY] The file descriptor for this port is configured for CCITT mode access. Hardware flow control is not allowed on CCITT mode devices.

AUTHOR

`termiox` was developed by HP and AT&T.

FILES

Files in or under `/dev/tty*`.

SEE ALSO

`ioctl(2)`, `termio(7)`, `modem(7)`.

NAME

timod - STREAMS module for converting ioctl() calls into Transport Interface messages

DESCRIPTION

The **timod** module is a STREAMS module that converts **ioctl()** calls from a transport user supporting the Transport Interface (TI) into messages that a transport protocol provider supporting TI can consume. This allows the user to initiate certain TI functions as atomic operations. This release of HP-UX no longer automatically pushes **timod** whenever a **t_open(3)** is performed. The TLI and XTI libraries have been modified to no longer require this module to perform the atomic operations described within this man page. Binary compatibility is not a problem since the module will still exist within the kernel. But, any application which is recompiled and expects the module to be automatically pushed, may not work without code modification.

The user places and removes the **timod** module on a device stream by calling the STREAMS **I_PUSH ioctl()** and **I_POP ioctl()** functions. (The TLI function **t_open()** pushes **timod** onto the device stream for the user.) The **timod** module should only be pushed onto streams which are terminated by transport providers which conform to the Transport Interface. **tirdwr(7)** is an alternative interface to **timod** which supports the **read()** and **write()** system calls. If **tirdwr** has been pushed onto the stream, the user should use the **I_POP ioctl** to remove the **tirdwr** module from the stream before pushing **timod**.

The **timod** module transparently passes any STREAMS messages that are not generated by the **ioctl()** commands described below to the neighboring module or driver. **timod** will act on an **I_STR ioctl()** whose **strioc1.ic_cmd** field is one of the values below. (See *streamio(7)* for a description of the **I_STR ioctl** and the **strioc1** structure.)

- | | |
|-------------------|--|
| TI_BIND | This TI command binds an address to the transport protocol provider. The STREAMS message that the module issues to the TI_BIND ioctl() call is equivalent to the TI message type T_bind_req . The STREAMS message that the module returns in response to the successful completion of the TI_BIND ioctl() call is equivalent to the TI message type T_bind_ack . |
| TI_UNBIND | This TI command unbinds an address from the transport protocol provider. The STREAMS message that the module issues to the TI_UNBIND ioctl() call is equivalent to the TI message type T_unbind_req . The STREAMS message that the module returns in response to the successful completion of the TI_UNBIND ioctl() call is equivalent to the TI message type T_ok_ack . |
| TI_GETINFO | This TI command gets the TI protocol-specific information from the transport protocol provider. The STREAMS message that the module issues to the TI_GETINFO ioctl() call is equivalent to the TI message type T_info_req . The STREAMS message that the module returns in response to the successful completion of the TI_GETINFO ioctl() call is equivalent to the TI message type T_info_ack . |
| TI_OPTMGMT | This TI command gets, sets, or negotiates TI protocol-specific options with the transport protocol provider. The STREAMS message that the module issues to the TI_OPTMGMT ioctl() call is equivalent to the TI message type T_optmgmt_req . The STREAMS message that the module returns in response to the successful completion of the TI_OPTMGMT ioctl() call is equivalent to the TI message type T_optmgmt_ack . |

RETURN VALUES

If the **timod** module returns an error for an **ioctl()** call, the lower 8 bits of the return value will be one of the TI error codes defined in the **<tiuser.h>** header file. If the TI error is of the type **TSYERR**, then the second 8 bits of the return value will contain an error as defined in the **<errno.h>** header file. The STREAMS message that the module issues when an **ioctl()** call results in an error is equivalent to the TI message type **T_error_ack**.

FILES

- | | |
|-------------------------|---|
| <xti.h> | defines the error codes for XTI functions. |
| <tiuser.h> | defines the error codes for TI functions. |
| <tihdr.h> | defines the message types for TI functions. |
| <errno.h> | defines the error codes for system errors. |

SEE ALSO

ioctl(2), t_open(3), streamio(7), tirdwr(7).


t

NAME

tirdwr - STREAMS module for reads and writes by Transport Interface users

DESCRIPTION

The **tirdwr** module is a STREAMS module that provides a transport user supporting the Transport Interface (TI) with an alternate interface to a transport protocol provider supporting TI. This alternate interface allows the transport user to communicate with the transport protocol provider using the **read()** and **write()** functions. It can also continue to use the **putmsg()** and **getmsg()** functions, but these functions will only transfer data messages between the user process and device stream. **getpmsg()** and **putpmsg()** should not be used with **tirdwr**.

The user places the **tirdwr** module on a device stream by calling the STREAMS **I_PUSH ioctl()** function. **tirdwr** is an alternative interface to **timod(7)**. If **timod** has been pushed onto the stream, the user should use the **I_POP ioctl** to remove the **timod** module from the stream before pushing **tirdwr**. The **tirdwr** module should only be pushed onto streams which are terminated by transport providers which conform to the Transport Interface. Once the module has been pushed on the device stream the user cannot make further calls to TI functions. If the user attempts to do this, an error occurs on the stream. After the error is detected, subsequent calls fail with **errno** set to [EPROTO]. The user removes the **tirdwr** module from a device stream by calling the STREAMS **I_POP ioctl()** function.

Module Behavior When Pushed and Popped

When the **tirdwr** module is pushed on a device stream, it checks any existing messages that are destined for the user to determine their message type. If existing messages are regular data messages, it forwards the messages to the user. It ignores any messages related to process management, such as messages that generate signals to the user. If any other messages are present, it returns an error to the user request with **errno** set to [EPROTO].

When the **tirdwr** module is popped from a device stream, it checks whether an orderly release indication has been previously received from the transport protocol provider. If an orderly release indication was received, it sends an orderly release request to the remote side of the transport connection. The **tirdwr** module also acts this way when the device stream is closed.

Module Behavior for Reads and Writes

When the **tirdwr** module receives messages from the transport protocol provider that do not contain a control part (see the **putmsg(2)** and **getmsg(2)** reference pages), it transparently passes the messages to its upstream neighbor. The exception is for zero-length data messages, where the module frees the message and does not pass them to its upstream neighbor.

When the module receives messages from the transport protocol provider that contain a control part, it takes one of the following actions:

For data messages with a control part, it removes this part, then passes the message to its upstream neighbor.

For messages that represent expedited data, it generates an error. Further system calls will fail with **errno** set to [EPROTO].

For messages that represent an orderly release indication from the transport protocol provider, it generates a zero-length data message, indicating the End-of-File (EOF), and sends this message upstream to the reading process. The original message containing the orderly release indication is freed.

For messages that represent an abortive disconnect indication from the transport protocol provider, it causes all further **write()** and **putmsg()** calls to fail with **errno** set to [ENXIO]. Subsequent **read()** and **getmsg()** calls will return zero-length data messages indicating the End-of-File (EOF), once all previous data has been read.

For all other messages, it generates an error, and further calls will fail with **errno** set to [EPROTO].

SEE ALSO

getmsg(2), **putmsg(2)**, **read(2)**, **write(2)**, **t_open(3)**, **streamio(7)**, **timod(7)**.

NAME

tty - controlling terminal interface

DESCRIPTION

The file `/dev/tty` is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that need to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

FILES

`/dev/tty`
`/dev/tty*`

SEE ALSO

`termio(7)`.

STANDARDS CONFORMANCE

`tty`: AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1

NAME

UDP - Internet User Datagram Protocol

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);
```

DESCRIPTION

UDP is a simple, unreliable datagram protocol used to support the **SOCK_DGRAM** socket type for the internet protocol family. UDP sockets are connectionless, and are normally used with the **sendto()** and **recvfrom()** calls (see *send(2)* and *recv(2)*). The **connect()** call can also be used to simulate a connection (see *connect(2)*). When used in this manner, it fixes the destination for future transmitted packets (in which case the **send()** or **write()** system calls can be used), as well as designating the source from which packets are received. The **recv()** and **read()** calls can be used at any time if the source of the message is unimportant.

UDP address formats are identical to those used by TCP. In particular, UDP requires a port identifier in addition to the normal Internet address format. Note that the UDP port domain is separate from the TCP port domain (in other words, a UDP port cannot be connected to a TCP port).

The maximum message size for a UDP datagram socket is limited by the lesser of the maximum size of an IP datagram and the size of the UDP datagram socket buffer. The send and receive buffers for UDP sockets can be altered to a maximum value of 262142 bytes by using the **SO_SNDBUF** and **SO_RCVBUF** options of the **setsockopt()** system call. However, the maximum size of an IP datagram limits the maximum message size of a UDP message to 65507 bytes. Therefore, using the maximum socket buffer size will allow multiple maximum-sized messages to be placed on the send queue. The default inbound and outbound message size limit for a UDP datagram socket is 65535 bytes. The maximum message size for a UDP broadcast is limited by the MTU size of the underlying link.

ERRORS

One of the following errors may be returned in **errno** if a socket operation fails. For a more detailed list of errors, see the man pages for specific system calls.

[EISCONN]	Attempt to send a datagram with the destination address specified, when the socket is already connected.
[ENOBUFS]	No buffer space is available for an internal data structure.
[EADDRINUSE]	Attempt to create a socket with a port which has already been allocated.
[EADDRNOTAVAIL]	Attempt to create a socket with a network address for which no network interface exists.

AUTHOR

The socket interfaces to UDP were developed by the University of California, Berkeley.

SEE ALSO

ndd(1M), getsockopt(2), recv(2), send(2), socket(2), t_open(3), inet(7F), socket(7),

RFC 768 User Datagram Protocol
RFC 1122 Requirements for Internet hosts

u

NAME

UNIX - local communication domain protocol

SYNOPSIS

```
#include <sys/types.h>
#include <sys/un.h>
```

DESCRIPTION

The local communication domain protocol, commonly referred to in the industry as the **Unix domain protocol**, utilizes the path name address format and the **AF_UNIX** address family. This protocol can be used as an alternative to the Internet protocol family (TCP/IP or UDP/IP) for communication between processes executing on the same node. It has a significant throughput advantage when compared with local IP loop-back, due primarily to its much lower code execution overhead. Data is looped back at the protocol layer (OSI Level 4), rather than at the driver layer (OSI Level 2).

Only **SOCK_STREAM** is supported in the **AF_UNIX** address family.

The HP-UX implementation of the local communication domain protocol does not support the **MSG_OOB** flag in **recv()** (see *recv(2)*) and **send()** (see *send(2)*).

Addressing

AF_UNIX socket addresses are path names. They are limited to 92 bytes in length, including a terminating null byte. Calls to **bind()** to an **AF_UNIX** socket utilize an addressing structure called **struct sockaddr_un** (see *bind(2)*). Pointers to this structure should be used in all **AF_UNIX** socket system calls wherever they require a pointer to a **struct sockaddr**.

The include file **<sys/un.h>** defines this addressing structure. Within this structure are two notable fields. The first is *sun_family*, which must be set to **AF_UNIX**. The next is *sun_path*, which is the null-terminated character string that specifies the path name of the file associated with the socket (for example, */tmp/mysocket*).

Only the passive (listening) socket must bind to an address. The active socket connects to that address, but it does not need an address of its own.

For additional information on using **AF_UNIX** sockets for interprocess communication, refer to the BSD Sockets Interface Programmer's Guide.

Socket Buffer Size

For stream and datagram sockets, the maximum send and receive buffer size is 262142 bytes. The default buffer size is 32768 bytes. The send and receive buffer sizes can be altered by using the **SO_SNDBUF** and **SO_RCVBUF** options of the **setsockopt()** system call. Refer to *getsockopt(2)* for details.

AUTHOR

AF_UNIX was developed by the University of California, Berkeley.

SEE ALSO

getsockopt(2), *socket(2)*.

NAME

vxfsio - VxFS-file-system control functions

SYNOPSIS

```
#include <sys/types.h>
#include <sys/fs/vx_ioctl.h>

int ioctl(int fildes, int cmd, ... /* arg */);
```

DESCRIPTION

The **vxfs** *ioctl*(2) enhancements provide for extended control over open files.

The argument **fildes** is an open file descriptor.

The data type and value of **arg** vary with the type of command indicated by **cmd**. Unless specified, **arg** is treated as an **int** type. The symbolic names for commands and file status flags are defined by the **sys/fs/vx_ioctl.h** header file.

The enhancements available are:

VX_SETCACHE

Set caching advisories. These advisories allow an application to indicate to the file system which forms of caching would be most advantageous.

NOTE: **VX_SETCACHE** is available with the VxFS Advanced package only.

The values for **arg** are such that multiple advisories may be set by combining values with bitwise **OR** operations. The possible values for **arg** are

VX_RANDOM

Indicates that the file is being accessed randomly. Read-ahead should not be performed.

VX_SEQ

Indicates that the file is being accessed sequentially. Maximum read-ahead should be performed.

VX_DIRECT

Indicates that data associated with read and write operations is to be transferred directly to or from the user supplied buffer, without being cached. When this options is enabled, all I/O operations must begin on block boundaries and must be a multiple of the block size in length. The buffer supplied with the I/O operations must be aligned to a page boundary.

If an I/O request fails to meet alignment criteria, or the file is currently being accessed for mapped I/O, the I/O request will be performed as a data synchronous I/O operation.

VX_NOREUSE

Indicates that buffered data does not need to be retained in anticipation of further use by the application.

VX_DSYNC

Indicates that data synchronous I/O mode is desired. In data synchronous I/O mode, a write operation returns to the caller after the data has been transferred to external media, but the inode is not updated synchronously if only the times in the inode need to be updated.

Only one of **VX_RANDOM**, **VX_SEQ**, or **VX_DIRECT** may be specified. The **VX_NOREUSE** and **VX_DSYNC** options may not be used in conjunction with **VX_DIRECT**. The caching advisories for a file are maintained on a per-file basis. Changes made to the advisories by a process affect I/O operations by all processes currently accessing the file.

The **VX_SETCACHE** *ioctl* returns a 0 if the caching advisories are successfully set. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_GETCACHE

Get caching advisories in effect for the file. The argument **arg** should be a pointer to an **int**.

The **VX_GETCACHE** *ioctl* returns a 0 if the caching advisories are successfully obtained and the advisories are returned in **arg**. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_SETEXT

Set extent information.

NOTE: **VX_SETEXT** is available with the VxFS Advanced package only.

The extent information is set according to the parameters specified by **arg**. The argument **arg** points to a structure of type **vx_ext** defined in **sys/fs/vx_ioctl.h**, which contains the following members:

```
off_t ext_size;    /* extent size in fs blocks */
off_t reserve;    /* space reservation in fs blocks */
int   a_flags;    /* allocation flags */
```

The **ext_size** element is used to request a fixed extent size, in file system blocks, for the file. If a fixed extent size is not required, zero should be used to allow the default allocation policy to be used. Changes to the fixed extent size made after the file contains indirect blocks have no effect unless all current indirect blocks are freed via file truncation and/or reservation deallocation.

The **reserve** element is used to set the amount of space, in file system blocks, preallocated to the file. If the **reserve** amount is greater than the current reservation, the allocation for the file is increased to match the **reserve** amount. If the **reserve** amount is less than the current reservation, the allocation is decreased. The allocation will not be reduced to less than the current file size.

File reservation cannot be increased beyond the *ulimit(2)* of the requesting process. However, an existing reservation will not be trimmed to the requesting process's *ulimit(2)*. Reservation of space for existing sparse files will not cause blocks to be allocated to fill in the holes, but will only allocate blocks after the end of the file. Thus, it's possible to have a larger reservation for a file than blocks in the file.

The reservation amount is independent of file size since reservation is used to preallocate space for a file.

The **a_flags** element is used to indicate the type of reservation required. The choices are:

VX_NOEXTEND

The file may not be extended once the current reservation is exceeded. The reservation may be increased if necessary by another invocation of the *ioctl*, but the file will not be automatically extended.

VX_TRIM

The reservation for the file is to be trimmed to the current file size upon last close by all processes that have the file open.

VX_CONTIGUOUS

The reservation must be allocated contiguously (as a single extent). **ext_size** will become the fixed extent size for subsequent allocations, but has no effect on this one. The reservation will fail if the file has gone into indirect extents, unless the amount of space requested is the same as the indirect extent size. If the contiguous allocation request is done on an empty file, this will not happen.

VX_ALIGN

Align all new extents on an **ext_size** boundary relative to the starting block of an allocation unit. If **VX_CONTIGUOUS** is also set, the single extent allocated during this invocation is not subject to the alignment restriction.

VX_NORESERVE

The reservation is to be made as a non-persistent allocation to the file. The on-disk inode will not be updated with the reservation information so that the reservation will not survive a system crash. The reservation is associated with the file until the close of the file. The reservation is trimmed to the current file size on close.

VX_CHGSIZE

The reservation is to be immediately incorporated into the file. The file's on-disk inode is updated with the size increased to include the reserved space. Unlike an *fcntl(2)* **F_FREESP** operation which "truncates up" (see *fcntl(2)*), the space included in the file is not initialized. This operation is restricted to users with appropriate privileges.

Write permission to a file is required to set extent information, but any process that can open the file can get the extent information. Extent information only applies to regular files. Only one set of extent information is kept per file. Only the **VX_ALIGN** and **VX_NOEXTEND** allocation flags are persistent attributes of the file. Other allocation flags may have persistent effects, but are not visible

as allocation flags. **VX_ALIGN**, **VX_NOEXTEND**, and **VX_TRIM** are the only flags visible through the **VX_GETTEXT** ioctl.

The **VX_SETTEXT** ioctl returns a 0 if the extent information is successfully set. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_GETTEXT

Get extent information. Return the extent information associated with **filides**. The argument **arg** points to a structure of type **vx_ext** as defined in **sys/fs/vx_ioctl.h**. Only persistent extent attributes are visible.

The **VX_GETTEXT** ioctl returns a 0 if the extent information is successfully obtained. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_GETFSOPT

Get file system options. The argument **arg** should be a pointer to an **int**. This command may be used by any user who can open the root inode on the file system. The options returned in **arg** are:

VX_FSO_BLKCLEAR

Indicates that any newly allocated blocks will be guaranteed to contain all zeros. (See the **blkclear** mount option of *mount_vxfs(1M)*).

VX_FSO_CACHE_CLOSESYNC

Indicates that any non-logged changes to the inode or data will be flushed to disk when the file is closed. (See the **mincache=closesync** mount option of *mount_vxfs(1M)*).

VX_FSO_CACHE_DIRECT

Indicates that any non-synchronous I/O will be handled as if the **VX_DIRECT** cache advisory had been set on the file. Also, any non-logged changes to the inode or data will be flushed to disk when the file is closed. (See the **mincache=direct** mount option of *mount_vxfs(1M)*).

VX_FSO_CACHE_DSYNC

Indicates that any writes that don't have either **O_SYNC** or the **VX_DIRECT** advisory set will be handled as if the **VX_DSYNC** advisory had been set on the file. Also, any non-logged changes to the inode or data will be flushed to disk when the file is closed. (See the **mincache=dsync** mount option of *mount_vxfs(1M)*).

VX_FSO_NODATAINLOG

Indicates that intent logging of user data for synchronous writes is disabled. (See the **nodatainlog** mount option of *mount_vxfs(1M)*).

VX_FSO_NOLOG

Indicates that intent logging of structural changes to the file system is disabled. (See the **nolog** mount option of *mount_vxfs(1M)*).

VX_FSO_OSYNC_CLOSESYNC

Indicates that any non-logged changes to the inode or data will be flushed to disk when a file accessed with **O_SYNC** is closed. (See the **convosync=closesync** mount option of *mount_vxfs(1M)*).

VX_FSO_OSYNC_DIRECT

Indicates that any **O_SYNC** I/O will be handled as if the **VX_DIRECT** cache advisory had been set on the file instead. Also, any non-logged changes to the inode or data will be flushed to disk when a file accessed with **O_SYNC** is closed. (See the **convosync=direct** mount option of *mount_vxfs(1M)*).

VX_FSO_OSYNC_DSYNC

Indicates that any **O_SYNC** writes will be handled as if the **VX_DSYNC** cache advisory had been set on the file instead. Also, any non-logged changes to the inode or data will be flushed to disk when a file accessed with **O_SYNC** is closed. (See the **convosync=dsync** mount option of *mount_vxfs(1M)*).

VX_FSO_SNAPPED

Indicates that a snapshot backup of this file system is being maintained.

VX_FSO_SNAPSHOT

Indicates that this file system is a snapshot backup of another file system.

VX_FSO_VJFS

Indicates that the VxFS Advanced package is not installed.

The **VX_GETFSOPT** ioctl returns a 0 if the file system options are successfully obtained. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_FREEZE

Sync then freeze the file system. Once frozen, all further operations against the file system block until a **VX_THAW** operation is received. The argument **arg** is a timeout value expressed in seconds. If a **VX_THAW** operation is not received within the specified timeout interval, the file system will perform a **VX_THAW** operation automatically.

This command may only be used by a user with appropriate privilege, on the root directory of the file system.

The **VX_FREEZE** ioctl returns a 0 if the file system is successfully frozen. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC.

VX_THAW

Unblock a file system that has been frozen by a **VX_FREEZE** operation. The argument **arg** should be NULL. The process that is to issue a **VX_THAW** operation must have the root directory of the file system open, and must ensure that it does not access the file system after the file system has been frozen, to ensure that the process itself does not block.

This command may only be used by a user with appropriate privilege, on the root directory of the file system.

The **VX_THAW** ioctl returns a 0 if the file system is successfully unfrozen. If the operation fails, the return value is -1 and the external variable **errno** will be a general DIAGNOSTIC or one of the following:

DIAGNOSTICS

The following values are returned in **errno** upon operation failures:

EACCESS	The calling process does not have write access to the file specified by files .
EAGAIN	The file system is not currently frozen.
EFBIG	An attempt was made to reserve space larger than the maximum file size limit for this process.
EINVAL	The command or argument is invalid.
EPERM	The process does not have appropriate privilege.
ENODEV	The file specified by files is not the root directory of a vxfs file system.
EROFS	The file system is mounted read-only.
EIO	An I/O error occurred while attempting to perform the operation.
ENOSPC	Requested space could not be obtained.
EFAULT	An address specified by an argument is invalid.

NOTES

Under certain circumstances, *fsadm*(1M) may reorganize the extent map of a file in such a way as to make it less contiguous. However, it will not change the geometry of a file that has a fixed extent size.

SEE ALSO

mount_vxfs(1M), getrlimit(2), ioctl(2), ulimit(2).

NAME

xopen_networking - X/Open Networking Interfaces

DESCRIPTION

X/Open has defined **XTI**, **Sockets**, and **IP Address Resolution** interfaces in *X/Open CAE Specification, Networking Services, Issue 4* (UNIX 95) and *X/Open CAE Specification, Networking Services, Issue 5* (UNIX 98). For a detailed description of these interfaces, please refer to the above specifications or to the book *Go Solo*, which is published by Prentice Hall. *Go Solo* also includes a CD ROM which contains the above specification to other *SPEC 1170* man pages.

COMPILING ENVIRONMENT

Using any of the Networking Services(**XTI**, **sockets**) requires `_XOPEN_SOURCE` to be defined and `_XOPEN_SOURCE_EXTENDED` to be defined with the value 1.

The `_XOPEN_SOURCE` macro may be defined automatically by the compilation environment, but to ensure portability the application should define the macro either on the compilation command line, or at the beginning of each source module prior to the inclusion of any headers. The `_XOPEN_SOURCE_EXTENDED` macro will not be automatically defined by the compilation environment.

The `c89` and `cc` utilities recognize the additional `-l` operand for standard libraries:

`-l xnet` This operand makes visible all functions referenced in the above specification.

AUTHOR

X/Open XTI, Sockets, and IP Address Resolution interfaces were developed by X/Open Company Limited.

SEE ALSO

XTI library functions:

`t_accept(3)`, `t_alloc(3)`, `t_bind(3)`, `t_close(3)`, `t_connect(3)`, `t_error(3)`, `t_free(3)`, `t_getinfo(3)`, `t_getprotaddr(3)`, `t_getstate(3)`, `t_listen(3)`, `t_look(3)`, `t_open(3)`, `t_optmgmt(3)`, `t_rcv(3)`, `t_rcvconnect(3)`, `t_rcvdis(3)`, `t_rcvrel(3)`, `t_rcvudata(3)`, `t_rcvuderr(3)`, `t_snd(3)`, `t_snddis(3)`, `t_sndrel(3)`, `t_sndudata(3)`, `t_strerror(3)`, `t_sync(3)`, `t_unbind(3)`.

Sockets Interfaces:

`accept(2)`, `bind(2)`, `close(2)`, `connect(2)`, `fcntl(2)`, `fgetpos(3S)`, `fsetpos(3S)`, `ftell(3S)`, `getpeername(2)`, `getsockname(2)`, `getsockopt(2)`, `listen(2)`, `lseek(2)`, `poll(2)`, `read(1)`, `recv(2)`, `recvfrom(2)`, `recvmsg(2)`, `select(2)`, `send(2)`, `sendmsg(2)`, `sendto(2)`, `setsockopt(2)`, `shutdown(2)`, `socket(2)`, `socketpair(2)`, `write(1)`.

IP Address Resolution Interfaces:

`endhostent(3N)`, `endnetent(3N)`, `endprotoent(3N)`, `endservent(3N)`, `gethostbyaddr(3N)`, `gethostname(3N)`, `getnetbyaddr(3N)`, `getprotobynumber(3N)`, `getservbyport(3N)`, `htonl(3N)`, `inet_addr(3N)`, `ntohl(3N)`, `sethostent(3N)`, `setnetent(3N)`, `setprotoent(3N)`, `setservent(3N)`.



X

Section 9

Glossary

Section 9
Glossary

NAME

intro - introduction to glossary section

DESCRIPTION

This section contains a glossary of common HP-UX terms. References to other HP-UX documentation are included as appropriate. Entities in *italics* with a following parenthesized roman number, such as *sh*(1), *wait*(2), or *fopen*(3S) refer to entries in the other sections of this manual. Items in **bold face** refer to other entries in this glossary. Any italicized manual names refer to separate manuals that are either included with your system or available separately.

The definitions specifically reflect the HP-UX operating system, although some terms and definitions are also derived from those in the emerging IEEE POSIX standards and the *X/Open Portability Guide*. Differences in wording exist to more specifically reflect the characteristics of the HP-UX system.

SEE ALSO

Introduction(9)

NAME

glossary - a description of common HP-UX terms

DESCRIPTION

HP-UX and other UNIX-like systems use a specialized vocabulary in which certain words and terms have very specific meanings. This glossary is intended as an aid in promoting exactness in use of these specialized terms whose meanings sometimes differ from those that might be encountered in other environments.

GLOSSARY ENTRIES

- . (dot)** A special file name that refers to the **current directory**. It can be used alone or at the beginning of a directory path name. See also **path name resolution**. The **dot** also functions as a special command in the Bourne and Korn shells, and has special meaning in text editors and formatters, in parsing regular expressions and in designating file names.
- .. (dot-dot)** A special file name that refers to the **parent directory**. If it begins a **path name**, **dot-dot** refers to the parent of the current directory. If it occurs in a path name, **dot-dot** refers to the parent directory of the directory preceding **dot-dot** in the path name string. As a special case, **dot-dot** refers to the current directory in any directory that has no parent (most often, the **root directory**). See also **path name resolution**.
- .o (dot-oh)** The suffix customarily given to a relocatable object file. The term **dot-oh file** is sometimes used to refer to a relocatable object file. The format of such files is sometimes called **dot-oh format**. See *a.out(4)*.

a.out The name customarily given to an executable object code file on HP-UX. The format is machine-dependent, and is described in *a.out(4)* for each implementation. Object code that is not yet linked has the same format, but is referred to as a **.o (dot-oh)** file. **a.out** is also the default output file name used by the linker, *ld(1)*.

absolute path name

A path name beginning with a slash (/). It indicates that the file's location is given relative to the **root directory** (/), and that the search begins there.

access The process of obtaining data from or placing data in storage, or the right to use system resources. Accessibility is governed by three process characteristics: the effective user ID, the effective group ID, and the group access list. The *access(2)* system call determines accessibility of a file according to the bit pattern contained in its *amode* parameter, which is constructed to read, write, execute or check the existence of a file. The *access(2)* system call uses the **real user ID** instead of the **effective user ID** and the **real group ID** instead of the **effective group ID**.

access groups

The group access list is a set of **supplementary group IDs** used in determining resource accessibility. Access checks are performed as described below in **file access permissions**.

access mode An access mode is a form of access permitted to a file. Each implementation provides separate read, write, and execute/search access modes.

address A number used in information storage or retrieval to specify and identify memory location. An **address** is used to mark, direct, indicate destination, instruct or otherwise communicate with computer elements.

In mail, **address** is a data structure whose format can be recognized by all elements involved in transmitting information. On a local system, this might be as simple as the user's **login** name, while in a networked system, **address** specifies the location of the resource to the network software.

In a text editor (such as **vi**, **ex**, **ed**, or **sed**), an **address** locates the line in a file on which a given instruction is intended.

For **adb**, the **address** specifies at what assembly-language instruction to execute a given command.

In disk utilities such as **fsdb**, **address** might refer to a raw or **block special file**, the **inode** number, **volume header**, or other file attribute.

In the context of peripheral devices, **address** refers to a set of values that specify the location of an I/O device to the computer. The exact details of the formation of an address differ between systems. On Series 700 systems, the address consists of up to two elements:

the **select code**, and the **function number**.

address space

The range of memory locations to which a process can refer.

affiliation See **terminal affiliation**.

appropriate privileges

Each implementation provides a means of associating privileges with a process for function calls and function call options requiring special privileges. In the HP-UX system, **appropriate privileges** refers either to superuser status or to a privilege associated with privilege groups (see *setprivgrp(1M)*).

archive A file comprised of the contents of other files, such as a group of object files (that is, **.o**) used by the linker, *ld(1)*). An archive file is created and maintained by *ar(1)* or similar programs, such as *tar(1)* or *cpio(1)*. An **archive** is often called a **library**.

ASCII An acronym for American Standard Code for Information Interchange. ASCII is the traditional System V coded character set and defines 128 characters, including both control characters and graphic characters, each of which is represented by 7-bit binary values ranging from 0 through 127 decimal.

background process group

Any process group that is a member of a session which has established a connection with a controlling terminal that is not in the foreground process group.

backup The process of making a copy of all or part of the file system in order to preserve it, in case a system crash occurs (usually due to a power failure, hardware error, etc.). This is a highly recommended practice.

block (1) The fundamental unit of information HP-UX uses for access and storage allocation on a mass storage medium. The size of a block varies between implementations and between file systems. In order to present a more uniform interface to the user, most system calls and utilities use **block** to mean 512 bytes, independent of the actual block size of the medium. This is the meaning of **block** unless otherwise specified in the manual entry.

(2) On media such as 9-track tape that write variable length strings of data, the size of those strings. **Block** is often used to distinguish from **record**; a block contains several records, whereas the number of records denotes the blocking factor.

block special file

A special file associated with a mass storage device (such as a hard disk or tape cartridge drive) that transfers data in multiple-byte blocks, rather than by series of individual bytes (see **character special file**). **Block special files** can be mounted. A **block special file** provides access to the device where hardware characteristics of the device are not visible.

boot, boot-up The process of loading, initializing, and running an operating system.

boot area A portion of a mass storage medium on which the volume header and a "bootstrap" program used in booting the operating system reside. The **boot area** is reserved exclusively for use by HP-UX.

boot ROM A program residing in ROM (Read-Only Memory) that executes each time the computer is powered up and is designed to bring the computer to a desired state by means of its own action. The first few instructions of a bootstrap program are sufficient to bring the remainder of the program into the computer from an input device and initiate functions necessary for computation. The function of the boot ROM is to run tests on the computer's hardware, find all devices accessible through the computer, and then load either a specified operating system or the first operating system found according to a specific search algorithm.

bus address A number which makes up part of the address HP-UX uses to locate a particular device. The **bus address** is determined by a switch setting on a peripheral device which allows the computer to distinguish between two devices connected to the same interface. A **bus address** is sometimes called a "device address".

character An element used for the organization, control, or representation of text. Characters include **graphic characters** and **control characters**.

character set A set of characters used to communicate in a native or computer language.

character special file

A special file associated with I/O devices that transfer data byte-by-byte. Other byte-mode I/O devices include printers, nine-track magnetic tape drives, and disk drives when accessed in “raw” mode (see **raw disk**). A **character special file** has no predefined structure.

child process A new process created by a pre-existing process via the *fork(2)* system call. The new process is thereafter known to the pre-existing process as its **child process**. The pre-existing process is the **parent process** of the new process. See **parent process** and **fork**.

clock tick A rate used within the system for scheduling and accounting. It consists of the number of intervals per second as defined by `CLK_TCK` that is used to express the value in type `clock_t`. `CLK_TCK` was previously known as the defined constant `HZ`.

coded character set

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its corresponding bit representation. **ASCII** is a **coded character set**.

collating element

The smallest entity used in collation to determine the logical ordering of strings (that is, the **collation sequence**). To accommodate native languages, a collating element consists of either a single character, or two or more characters collating as a single entity. The current value of the **LANG** environment variable determines the current set of collating elements.

collation The logical ordering of strings in a predefined sequence according to rules established by precedence. These rules identify a collation sequence among the collating elements and also govern the ordering of strings consisting of multiple collating elements, to accommodate native languages.

collation sequence

The ordering sequence applied to **collating elements** when they are sorted. To accommodate native languages, **collation sequence** can be thought of as the relative order of **collating elements** as set by the current value of the **LANG** environment variable. Characters can be omitted from the collation sequence, or two or more collating elements can be given the same relative order (see *string(3C)*).

command

A directive to perform a particular task. HP-UX commands are executed through a **command interpreter** called a **shell**. HP-UX supports several shells, including the Bourne shell (*sh-bourne(1)*), the POSIX shell (*sh-posix(1)*), the C shell (*csh(1)*), and the Korn shell (*ksh(1)*). See *sh(1)* for more information about supported shells. Most commands are carried out by an executable file, called a **utility**, which might take the form of a stand-alone unit of executable object code (a program) or a file containing a list of other programs to execute in a given order (a shell script). Scripts can contain references to other scripts, as well as to object-code programs. A typical **command** consists of the utility name followed by arguments that are passed to the utility. For example, in the command, “**ls mydirectory**”, “**ls**” is the utility name and “**mydirectory**” is an argument passed to the “**ls**” utility.

command interpreter

A program which reads lines of text from standard input (typed at the keyboard or read from a file), and interprets them as requests to execute other programs. A command interpreter for HP-UX is called a **shell**. See *sh(1)* and related manual entries.

Command Set 1980

See **CS/80**.

composite graphic symbol

A graphic symbol consisting of a combination of two or more other graphic symbols in a single character position, such as a diacritical mark and a basic letter.

control character

A character other than a graphic character that affects the recording, processing, transmission, or interpretation of text. In the **ASCII** character set, **control characters** are those in the range 0 through 31, and 127. Control characters can be generated by holding down

the control key (which may be labeled CTRL, CONTROL, or CNTL depending on your terminal), and pressing a character key (as you would use SHIFT). These two-key sequences are often written as, for example, Control-D, Ctrl-D, or ^D, where ^ stands for the control key.

controlling process

The session leader that establishes the connection to the **controlling terminal**. Should the terminal subsequently cease to be a controlling terminal for this session, the session leader ceases to be the controlling process.

controlling terminal

A terminal that is associated with a session. Each session can have at most one controlling terminal associated with it and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the foreground process group associated with the controlling terminal.

Coordinated Universal Time (UTC)

See **Epoch**.

CS/80, CS-80 A family of mass storage devices that communicate with the controlling computer by means of a series of commands and data transfer protocol referred to as the **CS/80** (Command Set 1980) command set. This command set was implemented in order to provide better forward/backward compatibility between models and generations of mass storage devices as technological advances develop. Some mass storage devices support only a subset of the full **CS/80** command set, and are usually referred to as **SS/80** (Subset 1980) devices.

crash The unexpected shutdown of a program or system. If the operating system crashes, this is a "system crash", and requires the system to be re-booted.

current directory

See **working directory**.

current working directory

See **working directory**.

daemon A process which runs in the background, and which is usually immune to termination instructions from a terminal. Its purpose is to perform various scheduling, clean-up, and maintenance jobs. *lpsched*(1M) is an example of a **daemon**. It exists to perform these functions for line printer jobs queued by *lp*(1). An example of a permanent **daemon** (that is, one that should never die) is *cron*(1M).

data encryption

A method for encoding information in order to protect sensitive or proprietary data. For example, HP-UX automatically encrypts all users' passwords. The encryption method used by HP-UX converts ASCII text into a base-64 representation using the alphabet `., /, 0-9, A-Z, a-z`. See *passwd*(4) for the numerical equivalents associated with this alphabet.

default search path

The sequence of directory prefixes that *sh*(1), *time*(1), and other HP-UX commands apply in searching for a file known by an relative path name (that is, a path name not beginning with a **slash** (/)). It is defined by the environment variable **PATH** (see *environ*(5)). *login*(1) sets **PATH** equal to `:/usr/bin`, which means that your working directory is the first directory searched, followed by `/usr/bin`. The search path can be redefined by modifying the value of **PATH**. This is usually done in `/etc/profile`, and/or in the `.profile` file found in the home directory.

delta A term used in the **Source Code Control System** (SCCS) to describe a unit of one or more textual changes to an **SCCS file**. Each time an SCCS file is edited, changes made to the file are stored separately as a **delta**. The *get*(1) command is then used to specify which deltas are to be applied to or excluded from the SCCS file, thus yielding a particular version of the file. Contrast this with the **vi** or **ed** editor, which incorporates changes into the file immediately, eliminating any possibility of obtaining a previous version of that file. A similar capability is provided by RCS files (see *rcsintro*(5)).

demon Improper spelling of the UNIX word **daemon**.

device A computer peripheral or an object that appears to an application as such.

d

device address

See **bus address**.

device file

See **special file**.

directory

A file that provides the mapping between the names of files and their contents, and is manipulated by the operating system alone. For every file name contained in a directory, that directory contains a pointer to the file's **inode**; The pointer is called a **link**. A file can have several links appearing anywhere on the same file system. Each user is free to create as many directories as needed (using *mkdir(1)*), provided that the **parent directory** of the new directory gives the permission to do so. Once a directory has been created, it is ready to contain ordinary files and other directories. An HP-UX directory is named and behaves exactly like an ordinary file, with one exception: no user (including the superuser) is allowed to write data on the directory itself; this privilege is reserved for the HP-UX operating system.

By convention, a directory contains at least two links, **.** and **..**, referred to as **dot** and **dot-dot** respectively. **Dot** refers to the directory itself and **dot-dot** refers to its **parent directory**. A directory containing only **.** and **..** is considered empty.

dot

See **.** (**dot**).

dot-dot

See **..** (**dot-dot**).

dot-oh

See **.o** (**dot-oh**).

dot-oh file

See **.o** (**dot-oh**).

dot-oh format

See **.o** (**dot-oh**).

downshifting The conversion of an uppercase character to its lowercase representation.

dynamic loader

A routine invoked at process startup time that loads shared libraries into a process' address space. The dynamic loader also resolves symbolic references between a program and the shared libraries, and initializes the shared libraries' linkage tables. See *dld.sl(5)* for details.

effective group ID

Every process has an **effective group ID** that is used to determine **file access permissions**. A process's **effective group ID** is determined by the file (command) that process is executing. If that file's set-group-ID bit is set (located in the mode of the file, see **mode**), the process's **effective group ID** is set equal to the file's group ID. This makes the process appear to belong to the file's group, perhaps enabling the process to access files that must be accessed in order for the program to execute successfully. If the file's set-group-ID bit is not set, the process's **effective group ID** is inherited from the process's parent. The setting of the process's **effective group ID** lasts only as long as the program is being executed, after which the process's effective group ID is set equal to its real group ID. See **group**, **real group ID**, and **set-group-ID bit**.

effective user ID

A process has an **effective user ID** that is used to determine **file access permissions** (and other permissions with respect to system calls, if the effective user ID is 0, which means superuser). A process's effective user ID is determined by the file (command) that process is executing. If that file's set-user-ID bit is set (located in the mode of the file, see **mode**), the process's effective user ID is set equal to the file's user ID. This makes the process appear to be the file's owner, enabling the process to access files which must be accessed in order for the program to execute successfully. (Many HP-UX commands which are owned by **root**, such as **mkdir** and **mail**, have their set-user-ID bit set so other users can execute these commands.) If the file's set-user-ID bit is not set, the process's effective user ID is inherited from that process's parent. See **real user ID** and **set-user-ID bit**.

end-of-file (EOF)

(1) The data returned when attempting to read past the logical end of a file via *stdio(3S)* routines. In this case, end-of-file is not properly a character.

(2) The ASCII character Ctrl-D.

(3) A character defined by *stty*(1) or *ioctl*(2) (see *termio*(7)) to act as end-of-file on your terminal. Usually this is Ctrl-D.

(4) The return value from *read*(2) that indicates end of data.

environment The set of defined shell variables (such as **EXINIT**, **HOME**, **PATH**, **SHELL**, **TERM**, and others) that define the conditions under which user commands run. These conditions can include user terminal characteristics, home directory, and default search path. Each shell variable setting in the current process is passed on to all **child processes** that are created, provided that each shell variable setting has been exported via the **export** command (see *sh*(1)). Unexported shell variable settings are meaningful only to the current process, and any child processes created get the default settings of certain shell variables by executing */etc/profile*, *\$HOME/.profile*, or *\$HOME/.login*.

EOF See **end-of-file**.

Epoch The time period beginning at 0 hours, 0 minutes, 0 seconds, **Coordinated Universal Time (UTC)** on January 1, 1970. Increments quantify the amount of time elapsed from the Epoch to the referenced time.

Leap seconds, which occur at irregular intervals, are not reflected in the count of seconds between the Epoch and the referenced time. (Fourteen leap seconds occurred in the years 1970 through 1988.)

FIFO special file

A type of **file**. Data written to a **FIFO** is read on a first-in-first-out basis. Other characteristics are described in *open*(2), *read*(2), *write*(2) and *lseek*(2).

file A stream of bytes that can be written to and/or read from. A **file** has certain attributes, including permissions and type. File types include **regular file**, **character special file**, **block special file**, **FIFO special file**, network special file, **directory**, and **symbolic link**. Every file must have a **file name** that enables the user (and many of the HP-UX commands) to refer to the contents of the file. The system imposes no particular structure on the contents of a file, although some programs do. Files can be accessed serially or randomly (indexed by byte offset). The interpretation of file contents and structure is up to the programs that access the file.

file access mode

A characteristic of an **open file description** that determines whether the described file is open for reading, writing, or both. (See *open*(2).)

file access permissions

Every file in the **file hierarchy** has a set of access permissions. These permissions are used in determining whether a process can perform a requested operation on the file (such as opening a file for writing). Access permissions are established when a file is created via the *open*(2) or *creat*(2) system calls, and can be changed subsequently through the *chmod*(2) call. These permissions are read by *stat*(2) or *fstat*(2).

File access controls whether a file can be read, written, or executed. Directory files use the execute permission to control whether or not the directory can be searched.

File access permissions are interpreted by the system as they apply to three different classes of users: the **owner** of the file, the users in the file's **group**, and anyone else ("other"). Every file has an independent set of access permissions for each of these classes. When an access check is made, the system decides if permission should be granted by checking the access information applicable to the caller.

Read, write, and execute/search permissions on a file are granted to a process if any of the following conditions are met:

- The process's **effective user ID** is superuser.
- The process's **effective user ID** matches the user ID of the owner of the file and the appropriate access bit of the **owner** portion (0700) of the file mode is set.
- The process's **effective user ID** does not match the user ID of the owner of the file, and either the process's **effective group ID** matches the group ID of the file, or the group ID of the file is in the process's group access list, and the appropriate access bit of the **group** portion (070) of the file mode is set.

- The process's **effective user ID** does not match the user ID of the owner of the file, and the process's **effective group ID** does not match the group ID of the file, and the group ID of the file is not in the process's group access list, and the appropriate access bit of the "other" portion (07) of the file mode is set.

Otherwise, the corresponding permissions are denied.

file descriptor

A small unique, per-process, nonnegative integer identifier that is used to refer to a file opened for reading and/or writing. Each **file descriptor** refers to exactly one **open file description**.

A **file descriptor** is obtained through system calls such as *creat(2)*, *fcntl(2)*, *open(2)*, *pipe(2)*, or *dup(2)*. The **file descriptor** is used as an argument by calls such as *read(2)*, *write(2)*, *ioctl(2)*, and *close(2)*.

The value of a **file descriptor** has a range from 0 to one less than the system-defined maximum. The system-defined maximum is the value **NOFILE** in `<sys/param.h>`.

file group class

A process is in the **file group class** of a file if the process is not the **file owner class** and if the **effective group ID** or one of the **supplementary group IDs** of the process matches the group ID associated with the file.

file hierarchy

The collection of one or more **file systems** available on a system. All **files** in these **file systems** are organized in a single hierarchical structure in which all of the nonterminal nodes are **directories**. Because multiple **links** can refer to the same **file**, the directory is properly described as a directed graph.

file name

A string of up to 14 bytes (or 255 bytes on file systems that support long file names) used to refer to an ordinary file, special file, or directory. The byte values NUL (null) and slash (/) cannot be used as characters in a file name. Note that it is generally unwise to use *, ?, ,, [, or] as part of file names because the shell attaches special meaning to these characters (see *sh(1)*, *csh(1)*, or *ksh(1)*). Avoid beginning a file name with -, +, or =, because to some programs, these characters signify that a command argument follows. A file name is sometimes called a path name component. Although permitted, it is inadvisable to use characters that do not have a printable graphic on the hardware you commonly use, or that are likely to confuse your terminal.

file name portability

File names should be constructed from the **portable file name character set** because the use of other characters can be confusing or ambiguous in certain contexts.

file offset

The file offset specifies the position in the file where the next I/O operation begins. Each **open file description** associated with either a regular file or special file has a **file offset**. There is no file offset specified for a **pipe** or **FIFO**.

file other class

A process is in the **file other class** if the process is not in the **file owner class** or **file group class**.

file owner class

A process is in the **file owner class** if the **effective user ID** of the process matches the user ID of the file.

file permission bits

See **permission bits**.

file pointer

A data element obtained through any of the *fopen(3S)* standard I/O library routines that "points to" (refers to) a file opened for reading and/or writing, and which keeps track of where the next I/O operation will take place in the file (in the form of a byte offset relative to the beginning of the file). After obtaining the file pointer, it must thereafter be used to refer to the open file when using any of the standard I/O library routines. (See *stdio(3S)* for a list of these routines.)

file serial number

A file-system-unique identifier for a given file, also known as the file's **inode number**. Each **file serial number** identifies exactly one **inode**. **File serial numbers** are not

necessarily unique across **file systems** in the **file hierarchy**.

file status flags

Part of an **open file description**. These flags can be used to modify the behavior of system calls that access the file described by the **open file description**.

file system

A collection of **files** and supporting data structures residing on a mass storage volume. A file system provides a name space for **file serial numbers** referring to those files. Refer to the System Administrator manuals supplied with your system for details concerning file system implementation and maintenance.

file times update

Each file has three associated time values that are updated when file data is accessed or modified, or when the file status is changed. These values are returned in the file characteristics structure, as described in `<sys/stat.h>`. For each function in HP-UX that reads or writes file data or changes the file status, the appropriate time-related files are noted as "marked-for-update". When an update point occurs, any marked fields are set to the current time and the update marks are cleared. One such update point occurs when the file is no longer open for any process. Updates are not performed for files on **read-only file systems**.

filter

A command that reads data from the standard input, performs a transformation on the data, and writes it to the standard output.

foreground process group

Each session that has established a connection with a controlling terminal has exactly one process group of the session as a foreground process group of that controlling terminal. The foreground process group has certain privileges when accessing its controlling terminal that are denied to background process groups. See *read(2)* and *write(2)*.

foreground process group ID

The process group ID of the foreground process group.

fork

An HP-UX system call (see *fork(2)*), which, when invoked by an existing process, causes a new process to be created. The new process is called the **child process**; the existing process is called the **parent process**. The child process is created by making an exact copy of the parent process. The parent and child processes are able to identify themselves by the value returned by their corresponding **fork** call (see *fork(2)* for details).

function number

On Series 700 systems, when two or more interfaces reside on a single interface card, each interface is treated as a separate **function** and is assigned a corresponding unique function number.

graphic character

A character other than a control character that has a visual representation when hand-written, printed, or displayed.

group

See **group ID**.

group ID

Associates zero or more users who must all be permitted to access the same set of files. The members of a group are defined in the files `/etc/passwd` and `/etc/loggingroup` (if it exists) via a numerical group ID that must be between zero and `UID_MAX`, inclusive. Users with identical group IDs are members of the same group. An ASCII group name is associated with each group ID in the file `/etc/group`. A group ID is also associated with every file in the **file hierarchy**, and the mode of each file contains a set of permission bits that apply only to this group. Thus, if you belong to a group that is associated with a file, and if the appropriate permissions are granted to your group in the file's mode, you can access the file. When the identity of a group is associated with a process, a group ID value is referred to as a **real group ID**, an **effective group ID**, a **supplementary group ID**, or a **saved group ID**. See also **privileged group** and **set-group-ID bit**.

group access list

A set of **supplementary group IDs** used in determining resource accessibility. Access checks are performed as described in **file access permissions**.

hierarchical directory

A directory (or file system) structure in which each directory can contain other directories as well as files.

home directory

The directory name given by the value of the environment variable **HOME**. When you first log in, *login*(1) automatically sets **HOME** to your **login directory**. You can change its value at any time. This is usually done in the **.profile** file contained in your **login directory**. Setting **HOME** does not affect your **login directory**; it simply gives you a convenient way of referring to what is probably your most commonly used directory.

host name

An ASCII string of at most 8 characters (of which only 6 are supported by all the various manufacturers' UNIX-like operating systems) which uniquely identifies an HP-UX system on a *uucp*(1) network. The **host name** for your system can be viewed and/or set with the *hostname*(1) command. Systems without a defined host name are described as "unknown" on the *uucp*(1) network. Do not confuse a host name with a **node name**, which is a string that uniquely identifies an HP-UX system on a Local Area Network (LAN). Although your host and node names may be identical, they are set and used by totally different software. See **node name**.

image

The current state of your computer (or your portion of the computer, on a multiuser system) during the execution of a command. Often thought of as a "snapshot" of the state of the machine at any particular moment during execution.

init

A **system process** that performs initialization, is the ancestor of every other process in the system, and is used to start **login** processes. **init** usually has a **process ID** of 1. See *init*(1M).

interleave factor

A number that determines the order in which sectors on a mass storage medium are accessed. It can be optimized to make data acquisition more efficient.

inode

An **inode** is a structure that describes a file and is identified in the system by a **file serial number**. Every file or directory has associated with it an **inode**. Permissions that specify who can access the file and how are kept in a 9-bit field that is part of the **inode**. The **inode** also contains the file size, the user and group ID of the file, the number of links, and pointers to the disk blocks where the file's contents can be found. Each connection between an **inode** and its entry in one or more directories is called a **link**.

inode number

See **file serial number**.

Internal Terminal Emulator (ITE)

The "device driver" code contained in the HP-UX kernel that is associated with the computer's built-in keyboard and display or with a particular keyboard and display connected to the computer, depending on the Series and Model of system processor. See **system console** and the System Administrator manuals supplied with your system for details.

internationalization

The concept of providing software with the ability to support the **native language**, **local customs**, and **coded character set** of the user.

interrupt signal

The signal sent by **SIGINT** (see *signal*(2)). This signal generally terminates whatever program you are running. The key which sends this signal can be redefined with *ioctl*(2) or *stty*(1) (see *termio*(7)). It is often the ASCII DEL (rubout) character (the DEL key) or the BREAK key. Ctrl-C is often used instead.

intrinsic

See **system call**.

I/O redirection

A mechanism provided by the HP-UX shell for changing the source of data for standard input and/or the destination of data for standard output and standard error. See *sh*(1).

ITE

See **Internal Terminal Emulator**.

job control

Job control allows users to selectively stop (suspend) execution of processes and continue (resume) their execution at a later time.

The user employs this facility via the interactive interface jointly supplied by the system terminal driver and certain shells (see *sh*(1)). The terminal driver recognizes a user-defined "suspend character", which causes the current foreground process group to stop and the user's job control shell to resume. The job control shell provides commands that

continue stopped process groups in either the foreground or background. The terminal driver also stops a background process group when any member of the background process group attempts to read from or write to the user's terminal. This allows the user to finish or suspend the **foreground process group** without interruption and continue the stopped **background process group** at a more convenient time.

See *stty*(1), *sh*(1), and related shell entries for usage and installation details, and the shell entries plus *signal*(2) and *termio*(7) for implementation details.

kernel The HP-UX operating system. The kernel is the executable code responsible for managing the computer's resources, such as allocating memory, creating processes, and scheduling programs for execution. The kernel resides in RAM (random access memory) whenever HP-UX is running.

LANG An environment variable used to inform a computer process of the user's requirements for **native language**, **local customs**, and **coded character set**.

library A file containing a set of subroutines and variables that can be accessed by user programs. Libraries can be either archives or shared libraries. For example, `/usr/lib/libc.a` and `/usr/lib/libc.sl` are libraries containing all functions of Section 2 and all functions of Section 3 that are marked (3C) and (3S) in the *HP-UX Reference Manual*. Similarly, `/usr/lib/libm.a` and `/usr/lib/libm.sl` are libraries containing all functions in Section 3 that are marked (3M) in the *HP-UX Reference Manual*. See *intro*(2) and *intro*(3).

LIF See **Logical Interchange Format**.

line A sequence of text characters consisting of zero or more nonnewline characters plus a terminating newline character.

link **Link** is a synonym for **directory entry**. It is an object that associates a file name with any type of file. The information constituting a **link** includes the name of the file and where the contents of that file can be found on a mass storage medium. One physical file can have several links to it. Several directory entries can associate names with a given file. If the links appear in different directories, the file may or may not have the same name in each. However, if the links appear in one directory, each link must have a unique name in that directory. Multiple links to directories are not allowed (except as created by a user with appropriate privileges). See *ln*(1), *link*(2), *unlink*(2), and **symbolic link**.

Also, to prepare a program for execution; see **linker**.

link count The number of directory entries that refer to a particular file.

linker A program that combines one or more object programs into one program, searches libraries to resolve user program references, and builds an executable file in `a.out` format. This executable file is ready to be executed through the program loader, *exec*(2). The linker is invoked with the *ld*(1) command. The linker is often called a **link editor**.

local customs The conventions of a geographical area or territory for such things as date, time and currency formats.

localization The process of adapting existing software to meet the local language, customs, and character set requirements of a particular geographical area.

Logical Interchange Format (LIF)

A standard format for mass storage implemented on many Hewlett-Packard computers to aid in media transportability. The *lif**(1) commands are used to perform various LIF functions.

login The process of gaining access to HP-UX. This consists of successful execution of the **login** sequence defined by *login*(1), which varies depending on the system configuration. It requests a **login** name and possibly one or more passwords.

login directory

The directory in which you are placed immediately after you log in. This directory is defined for each user in the file `/etc/passwd`. The shell variable **HOME** is set automatically to your **login directory** by *login*(1) immediately after you log in. See **home directory**.

magic number

The first word of an **a.out**-format or archive file. This word contains the system ID, which states what machine (hardware) the file will run on, and the file type (executable, sharable executable, archive, etc.).

major number

A number used exclusively to create special files that enable I/O to or from specific devices. This number indicates which device driver to use for the device. Refer to *mknod(2)* and the System Administrator manual supplied with your system for details.

message catalog

Program strings, such as program messages and prompts, are stored in a **message catalog** corresponding to a particular geographical area. Retrieval of a string from a **message catalog** is based on the value of the user's **LANG** environment variable (see **LANG**).

message queue identifier (msqid)

A unique positive integer created by a *msgget(2)* system call. Each **msqid** has a message queue and a data structure associated with it. The data structure is referred to as **msqid_ds** and contains the following members:

```
struct ipc_perm msg_perm; /* operation permission */
ushort msg_qnum;          /* number of msgs on q */
ushort msg_qbytes;        /* max number of bytes on q */
ushort msg_lspid;         /* pid of last msgsnd operation */
ushort msg_lrpid;         /* pid of last msgrcv operation */
time_t msg_stime;         /* last msgsnd time */
time_t msg_rtime;         /* last msgrcv time */
time_t msg_ctime;         /* last change time */
                          /* Times measured in secs since */
                          /* 00:00:00 GMT, Jan. 1, 1970 */
```

Message queue identifiers can be created using *ftok(3C)*.

msg_perm is a **ipc_perm** structure that specifies the message operation permission (see below). This structure includes the following members:

```
ushort cuid;             /* creator user id */
ushort cgid;             /* creator group id */
ushort uid;              /* user id */
ushort gid;              /* group id */
ushort mode;             /* r/w permission */
```

msg_qnum is the number of messages currently on the queue. **msg_qbytes** is the maximum number of bytes allowed on the queue. **msg_lspid** is the process id of the last process that performed a **msgsnd** operation. **msg_lrpid** is the process id of the last process that performed a **msgrcv** operation. **msg_stime** is the time of the last **msgsnd** operation, **msg_rtime** is the time of the last **msgrcv** operation, and **msg_ctime** is the time of the last *msgctl(2)* operation that changed a member of the above structure.

message operation permissions

In the *msgop(2)* and *msgctl(2)* system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has these permissions for an object is determined by the object's permission mode bits as follows:

```
00400    Read by user
00200    Write by user
00060    Read, Write by group
00006    Read, Write by others
```

Read and Write permissions on a **msqid** are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches **msg_perm.[c]uid** in the data structure associated with **msqid** and the appropriate bit of the "user" portion (0600) of **msg_perm.mode** is set.

- The process's effective user ID does not match `msg_perm.[c]uid` and either the process's effective group ID matches `msg_perm.[c]gid` or one of `msg_perm.[c]gid` is in the process's group access list and the appropriate bit of the "group" portion (00060) of `msg_perm.mode` is set.
- The process's effective user ID does not match `msg_perm.[c]uid` and the process's effective group ID does not match `msg_perm.[c]gid` and neither of `msg_perm.[c]gid` is in the process's group access list and the appropriate bit of the "other" portion (06) of `msg_perm.mode` is set.

Otherwise, the corresponding permissions are denied.

metacharacter

A character that has special meaning to the HP-UX shell, as well as to commands such as `ed`, `find`, and `grep` (see `ed(1)`, `find(1)`, and `grep(1)`). The set of metacharacters includes: `!`, `"`, `&`, `'`, `*`, `.`, `<`, `>`, `?`, `[`, `]`, `\`, and `|`. Refer to `sh(1)` and the related shell manual entries for the meaning associated with each. See also **regular expression**.

minor number

A number that is an attribute of special files, specified during their creation and used whenever they are accessed, to enable I/O to or from specific devices. This number is passed to the device driver and is used to select which device in a family of devices is to be used, and possibly some operational modes. The exact format and meaning of the **minor number** is both system and driver dependent. Refer to the System Administrator manuals supplied with your system for details.

On Series 700 systems, a **minor number** indicates the device address, function number, and driver-dependent bits. On Series 800 systems, a **minor number** is an index into a table in the **kernel**.

mode

A 16-bit word associated with every file in the file system, stored in the **inode**. The least-significant 12 bits of the **mode** determine the read, write, and execute permissions for the file owner, file group, and all others, and contain the set-user-ID, set-group-ID, and "sticky" (save text image after execution) bits. The least-significant 12 bits can be set by the `chmod(1)` command if you are the file's owner or the superuser. The sticky bit on a regular file can only be set by the superuser. These 12 bits are sometimes referred to as **permission bits**. The most-significant 4 bits specify the file type for the associated file and are set as the result of `open(2)` or `mknod(2)` system calls.

mountable file system

A removable blocked file system contained on some mass storage medium with its own root directory and an independent hierarchy of directories and files. See **block special file** and `mount(1M)`.

msqid

See **message queue identifier**.

multiuser state

The condition of the HP-UX operating system in which terminals (in addition to the system console) allow communication between the system and its users. By convention, multiuser run level is set at state 2, which is usually defined to contain all the terminal processes and **daemons** needed in a multiuser environment. Run levels are table driven, and are specified by `init(1M)`, which sets the run level by looking at the file `/etc/inittab`. Do not confuse the multiuser system with the multiuser state. A multiuser system is a system which can have more than one user actively communicating with the system when it is in the multiuser state. The multiuser state removes the single-user restriction imposed by the single-user state (see **single-user state**, `inittab(4)`).

native language

A computer user's spoken or written language, such as Chinese, Dutch, English, French, German, Greek, Italian, Katakana, Korean, Spanish, Swedish, Turkish, etc.

Native Language Support (NLS)

A feature of HP-UX that provides the user with internationalized software and the application programmer with tools to develop this software.

newline character

The character with an ASCII value of 10 (line feed) used to separate lines of characters. It is represented by `\n` in the C language and in various utilities. The terminal driver

normally interprets a carriage-return/line-feed sequence sent by a terminal as a single new-line character (but see *tty(7)* for full details)

NLS See **Native Language Support**.

NLSPATH An environment variable used to indicate the search path for message catalogs (see **message catalog**).

node name A string of up to 31 characters, not including control characters or spaces, that uniquely identifies a node on a Local Area Network (LAN). The **node name** for each system is set by the **npowerup** command, which is one of the commands supplied with optional LAN/9000 products. Do not confuse a node name with a **host name**, which is a string that uniquely identifies an HP-UX system on a UUCP network. Your node and host names can be identical, but they are used and set by totally different software. See **host name**, *LAN/9000 User's Guide*, and *LAN/9000 Node Manager's Guide*.

nonspacing characters

Characters, such as a diacritical mark or accents, that are used in combination with other characters to form composite graphic symbols commonly found in non-English languages.

open file A file that is currently associated with a file descriptor.

open file description

A record of how a process or a group of processes is accessing a file. Each **file descriptor** refers to exactly one **open file description**, but an **open file description** can be referred to by more than one file descriptor. The **file offset**, **file status flags**, and **file access modes** are attributes of an **open file description**.

ordinary file A type of HP-UX file containing ASCII text (e.g., program source), binary data (e.g., executable code), etc. Ordinary files can be created by the user through I/O redirection, editors, or HP-UX commands.

orphan process

A **child process** that is left behind when a **parent process** terminates for any reason. The **init** process (see *init(1M)*) inherits (that is, becomes the effective parent of) all orphan processes.

orphaned process group

A process group in which the parent of every member is either itself a member of the group or is not a member of the group's session.

owner The owner of a file is usually the creator of that file. However, the ownership of a file can be changed by the superuser or the current owner with the *chown(1)* command or the *chown(2)* system call. The file owner is able to do whatever he wants with his files, including remove them, copy them, move them, change their contents, etc. The owner can also change the files' modes.

parent directory

The directory one level above a directory in the **file hierarchy**. All directories except the **root directory** (/) have one (and only one) parent directory. The **root directory** has no parent. See also **dot** and **dot-dot**.

parent process

Whenever a new process is created by a currently-existing process (via *fork(2)*), the currently existing process is said to be the parent process of the newly created process. Every process has exactly one parent process (except the **init** process, see **init**), but each process can create several new processes with the *fork(2)* system call. The parent process ID of any process is the **process ID** of its creator.

parent process ID

A new process is created by a currently active process. The **parent process ID** of a process is the process ID of its creator for the lifetime of the creator. After the creator's lifetime has ended, the **parent process ID** is the process ID of **init**.

password A string of ASCII characters used to verify the identity of a user. Passwords can be associated with users and groups. If a user has a password, it is automatically encrypted and entered in the second field of that user's line in the */etc/passwd* file. A user can create or change his or her own password by using the *passwd(1)* command.

path name A sequence of directory names separated by slashes, and ending with any file name. All file names except the last in the sequence *must* be directories. If a path name begins with a **slash (/)**, it is an **absolute path name**; otherwise, it is a **relative path name**. A path name defines the path to be followed through the hierarchical file system in order to find a particular file.

More precisely, a path name is a null-terminated character string constructed as follows:

```
<path-name>::=<file-name>|<path-prefix><file-name>|/
<path-prefix>::=<rtprefix>|/<rtprefix>
<rtprefix>::=<dirname>/|<rtprefix><dirname>/
```

where <file-name> is a string of one or more characters other than the ASCII slash and null, and <dirname> is a string of one or more characters (other than the ASCII slash and null) that names a directory. File and directory names can consist of up to 14 characters on systems supporting short file names and up to 255 characters on systems supporting long file names.

A **slash (/)** by itself names the **root directory**. Two or more slashes in succession (///...) are treated as a single slash.

Unless specifically stated otherwise, the null or zero-length path name is treated as though it named a nonexistent file.

path name resolution

The process that resolves a path name to a particular file in a **file hierarchy**. Multiple path names can resolve to the same file, depending on whether resolution is sought in absolute or relative terms (see below). Each file name in the path name is located in the directory specified by its predecessor (for example, in the path name fragment **a/b**, file **b** is located in directory **a**). **Path name resolution** fails if this cannot be accomplished.

If the path name begins with a slash, the predecessor of the first file name in the path name is understood to be the **root directory** of the process, and the path name is referred to as an **absolute path name**. If the path name does not begin with a slash, the predecessor of the first file name of the path name is understood to be the current working directory of the process, and the path name is referred to as a **relative path name**. A path name consisting of a single slash resolves to the root directory of the process.

path prefix A **path name** with an optional ending **slash** that refers to a **directory**.

permission bits

The nine least-significant bits of a file's **mode** are referred to as file **permission bits**. These bits determine read, write, and execute permissions for the file's **owner**, the file's **group**, and all others. The bits are divided into three parts: owner, group and other. Each part is used with the corresponding file class of processes. The bits are contained in the file mode, as described in *stat(5)*. The detailed usage of the file permission bits in access decisions is described in **file access permissions**.

PIC See **position-independent code**.

pipe An interprocess I/O channel used to pass data between two processes. It is commonly used by the **shell** to transfer data from the standard output of one process to the standard input of another. On a command line, a pipe is signaled by a vertical bar (|). Output from the command to the left of the vertical bar is channeled directly into the standard input of the command on the right.

portable file name character set

The following set of graphical characters are portable across conforming implementations of IEEE Standard P1003.1:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
01234567890._-
```

The last three characters are the dot, underscore and hyphen characters, respectively. The hyphen should not be used as the first character of a portable file name.

position-independent code (PIC)

Object code that can run unmodified at any virtual address. Position-independent code can use PC-relative addressing modes and/or linkage tables. It is most often used in shared

libraries, in which case the linkage tables are initialized by the dynamic loader. Position-independent code is generated when the **+z** or **+Z** compiler option is specified.

privileged groups

A **privileged group** is a group that has had a **setprivgrp** (see *getprivgrp(2)*) operation performed on it, giving it access to some system calls otherwise reserved for the superuser. See **appropriate privileges**.

process

An invocation of a program, or the execution of an image (see **image**). Although all commands and utilities are executed within processes, not all commands or utilities have a one-to-one correspondence with processes. Some commands (such as **cd**) execute within a process, but do not create any new processes. Others (such as in the case of **ls** | **wc -l**) create multiple processes. Several processes can be running the same program, but each can be different data and be in different stages of execution. A process can also be thought of as an **address space** and single thread of control that executes within that address space and its required system resources. A **process** is created by another process issuing the *fork(2)* function. The process that issues *fork(2)* is known as the **parent process** and the new process created by the *fork(2)* as the **child process**.

process 1 See **init**.

process group

Each process in the system is a member of a **process group**. This grouping permits the signaling of related processes. A newly created process joins the process group of its creator.

process group ID

Each process group in the system is uniquely identified during its lifetime by a **process group ID**, a positive integer less than or equal to **PIC_MAX**. A **process group ID** cannot be reused by the system until the process group lifetime ends.

process group leader

A **process group leader** is a process whose process ID is the same as its process group ID.

process group lifetime

A period of time that begins when a **process group** is created and ends when the last remaining process in the group leaves the group, either due to process termination or by calling the *setsid(2)* or *setpgid(2)* functions.

process ID

Each active process in the system is uniquely identified during its lifetime by a positive integer less than or equal to **PID_MAX** called a **process ID**. A process ID cannot be reused by the system until after the process lifetime ends. In addition, if there exists a process group whose process group ID is equal to that process ID, the process ID cannot be reused by the system until the process group lifetime ends. A process that is not a system process shall not have a process ID of 1.

process lifetime

After a process is created with a *fork(2)* function, it is considered active. Its thread of control and **address space** exist until it terminates. It then enters an inactive state where certain resources may be returned to the system, although some resources, such as the **process ID** are still in use. When another process executes a *wait()*, *wait3()*, or *waitpid()* function (see *wait(2)*) for an inactive process, the remaining resources are returned to the system. The last resource to be returned to the system is the process ID. At this time, the lifetime of the process ends.

program

A sequence of instructions to the computer in the form of binary code (resulting from the compilation and assembly of program source).

prompt

The characters displayed by the **shell** on the terminal indicating that the system is ready for a command. The prompt is usually a dollar sign (\$) for ordinary users (% in the C shell) and a pound sign (#) for the superuser, but you can redefine it to be any string by setting the appropriate shell variable (see *sh(1)* and related entries). See also **secondary prompt**.

quit signal

The **SIGQUIT** signal (see *signal(2)*). The quit signal is generated by typing the character defined by the teletype handler as your quit signal. (See *stty(1)*, *ioctl(2)*, and *termio(7)*.) The default is the ASCII FS character (ASCII value 28) generated by typing Ctrl-\ . This signal usually causes a running program to terminate and generates a file containing the "core image" of the terminated process. The core image is useful for debugging purposes.

(Some systems do not support core images, and on those systems no such file is generated.)

radix character

The character that separates the integer part of a number from the fractional part. For example, in American usage, the **radix character** is a decimal point, while in Europe, a comma is used.

raw disk

The name given to a disk for which there exists a **character special file** that allows direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O call.

read-only file system

A characteristic of a **file system** that prevents file system modifications.

real group ID

A positive integer which is assigned to every user on the system. The association of a user and his or her **real group ID** is done in the file `/etc/passwd`. The modifier "real" is used because a user can also have an **effective group ID**. The real group ID can then be mapped to a group name in the file `/etc/group`, although it need not be. Thus, every user is a member of some group (which can be nameless), even if that group has only one member.

Every time a process creates a child process (via `fork(2)`), that process has a real group ID equal to the parent process's real group ID. This is useful for determining file access privileges within the process.

real user ID

A positive integer which is assigned to every user on the system. A real user ID is assigned to every valid **login** name in the file `/etc/passwd`. The modifier "real" is used because a user can also have an **effective user ID** (see **effective user ID**).

Every time a process creates a child process (via `fork(2)`), that process has a real user ID equal to the parent process's real user ID. This is useful for determining file access privileges within the process.

regular expression

A string of zero or more characters that selects text. All the characters contained in the string might be literal, meaning that the regular expression matches itself only; or one or more of the characters might be a **metacharacter**, meaning that a single regular expression could match several literal strings. Regular expressions are most often encountered in text editors (such as `ed(1)`, `ex(1)`, or `vi(1)`), where searches are performed for a specific piece of text, or in commands that were created to search for a particular string in a file (most notably `grep(1)`). Regular expressions are also encountered in the shell, especially when referring to file names on command lines.

regular file

A type of **file** that is a randomly accessible sequence of bytes, with no further structure imposed by the system. Its size can be extended. A regular file is also called an **ordinary file**.

relative path name

A **path name** that does not begin with a **slash** (/). It indicates that a file's location is given relative to your current **working directory**, and that the search begins there (instead of at the **root directory**). For example, `dir1/file2` searches for the directory `dir1` in your current working directory; then `dir1` is searched for the file `file2`.

root directory

(1) The highest level directory of the hierarchical file system, from which all other files branch. In HP-UX, the **slash** (/) character refers to the **root directory**. The root directory is the only directory in the file system that is its own **parent directory**.

(2) Each process has associated with it a concept of a root directory for the purpose of resolving path name searches for those paths beginning with **slash** (/). A process's root directory need not be the root directory of the root file system, and can be changed by the `chroot(1M)` command or `chroot(2)` system call. Such a directory appears to the process involved to be its own parent directory.

root volume

The mass storage volume which contains the boot area (which contains the HP-UX kernel) and the **root directory** of the HP-UX file system.

saved group ID

Every process has a saved group ID that retains the process's **effective group ID** from the last successful *exec(2)* or *setresgid()* (see *setresuid(2)*), or from the last superuser call to *setgid()* (see *setuid(2)*) or *setresuid(2)*. *setgid()* permits a process to set its effective group ID to this remembered value. Consequently, a process that executes a program with the set-group-ID bit set and with a group ID of 5 (for example) can set its effective group ID to 5 at any time until the program terminates. See *exec(2)*, *setuid(2)*, **saved user ID**, **effective group ID**, and **set-group-ID bit**. The saved group ID is also known as the **saved set-group-ID**.

saved process group ID

Every process has a saved process group ID that retains the process's group ID from the last successful *exec(2)*. See *setpgrp(2)*, *termio(7)*, and **process group ID**.

saved user ID

Every process has a **saved user ID** that retains the process's **effective user ID** from the last successful *exec(2)* or *setresuid(2)*, or from the last superuser call to *setuid(2)*. *setuid(2)* permits a process to set its effective user ID to this remembered value. Consequently, a process which executes a program with the set-user-ID bit set and with an owner ID of 5 (for example) can set its effective user ID to 5 at any time until the program terminates. See *exec(2)*, *setuid(2)*, **saved group ID**, **effective user ID**, and **set-user-ID bit**. The saved user ID is also known as the **saved set-user-ID**.

saved set-group-ID

See **saved group ID**.

saved set-user-ID

See **saved user ID**.

SCCS

See **Source Code Control System**.

Source Code Control System (SCCS)

A set of HP-UX commands that enables you to store changes to an **SCCS file** as separate "units" (called **deltas**). These units, each of which contains one or more textual changes to the file, can then be applied to or excluded from the SCCS file to obtain different versions of the file. The commands that make up SCCS are *admin(1)*, *cdc(1)*, *delta(1)*, *get(1)*, *prs(1)*, *rm(1)*, *sact(1)*, *sc(1)*, *scsdiff(1)*, *unget(1)*, *val(1)*, and *what(1)*.

SCCS file

An ordinary text file that has been modified so the **Source Code Control System (SCCS)** can be used with it. This modification is done automatically by the *admin(1)* command. See also **delta**.

secondary prompt

One or more characters that the shell prints on the display, indicating that more input is needed. This prompt is not encountered nearly as frequently as the shell's primary prompt (see **prompt**). When it occurs, it is usually caused by an omitted right quote on a string (which confuses the shell), or when you enter a shell programming language control-flow construct (such as a **for** construct) from the command line. By default, the shell's secondary prompt is the greater-than sign (>), but you can re-define it by setting the shell variable **PS2** appropriately in your **.profile** file. (The C shell has no secondary prompt.)

select code

On Series 700 systems, part of an **address** used for devices. Multiple peripherals connected to the same interface card share the same select code. On Series 700 systems, **select code** consists of the bus and slot numbers for a device, both of which are determined by the particular I/O slot in which the I/O card resides. All functions on a multifunction card share the same select code.

semaphore identifier (semid)

A unique positive integer created by a *semget(2)* system call. Each **semid** has a set of semaphores and a data structure associated with it. The data structure is referred to as **semid_ds** and contains the following members:

```
struct ipc_perm sem_perm; /* operation permission */
ushort sem_nsems;        /* number of sems in set */
time_t sem_otime;        /* last operation time */
time_t sem_ctime;        /* last change time */
/* Times measured in secs since */
/* 00:00:00 GMT, Jan. 1, 1970 */
```

Semaphore identifiers can be created using *flok*(3C).

sem_perm is a *ipc_perm* structure that specifies the semaphore operation permission (see below). This structure includes the following members:

```
ushort  cuid;    /* creator user id */
ushort  cgid;    /* creator group id */
ushort  uid;     /* user id */
ushort  gid;     /* group id */
ushort  mode;    /* r/a permission */
```

The value of **sem_nsems** is equal to the number of semaphores in the set. Each semaphore in the set is referenced by a positive integer referred to as a **sem_num**. **sem_num** values run sequentially from 0 to the value of **sem_nsems** minus 1. **sem_otime** is the time of the last *semop*(2) operation, and **sem_ctime** is the time of the last *semctl*(2) operation that changed a member of the above structure.

A semaphore is a data structure that contains the following members:

```
ushort  semval;   /* semaphore value */
short   sempid;   /* pid of last operation */
ushort  semncnt;  /* # awaiting semval > cval */
ushort  semzcnt;  /* # awaiting semval = 0 */
```

semval is a nonnegative integer. **sempid** is equal to the process ID of the last process that performed a semaphore operation on this semaphore. **semncnt** is a count of the number of processes that are currently suspended awaiting this semaphore's **semval** to become greater than its current value. **semzcnt** is a count of the number of processes that are currently suspended awaiting this semaphore's **semval** to become zero.

semaphore operation permissions

In the *semop*(2) and *semctl*(2) system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has these permissions for an object is determined by the object's permission mode bits as follows:

```
00400    Read by user
00200    Alter by user
00060    Read, Alter by group
00006    Read, Alter by others
```

Read and Alter permissions on a **semid** are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches **sem_perm.[c]uid** in the data structure associated with **semid** and the appropriate bit of the "user" portion (0600) of **sem_perm.mode** is set.
- The process's effective user ID does not match **sem_perm.[c]uid** and the appropriate bit of the "group" portion (060) of **sem_perm.mode** is set.
- The process's effective user ID does not match **sem_perm.[c]uid** and the process's effective group ID does not match **sem_perm.[c]gid** and neither of **sem_perm.[c]gid** is in the process's group access list and the appropriate bit of the "other" portion (06) of **sem_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

semid See **semaphore identifier**.

session Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership (see *setsid*(2)). A session can have multiple process groups (see *setpgid*(2)).

session leader

A process that has created a session (see *setsid*(2)).

session lifetime

The period between when a session is created and the end of the lifetime of all process

groups that remain as members of the session.

set-group-ID bit

A single bit in the mode of every file in the file system. If a file is executed whose **set-group-ID bit** is set, the **effective group ID** of the process which executed the file is set equal to the **real group ID** of the owner of the file. See also **group**.

set-user-ID bit

A single bit in the mode of every file in the file system. If a file is executed whose **set-user-ID bit** is set, the **effective user ID** of the process that executed the file is set equal to the **real user ID** of the owner of the file.

shared library

An executable file that can be shared between several different programs. Code from a shared library is not linked into the program by *ld(1)*, but is instead mapped into the process' address space at run time by the dynamic loader. Shared libraries must contain position-independent code, and are created by *ld(1)*. They typically have the file name suffix **.sl**.

shared memory identifier (shmid)

A unique positive integer created by a *shmget(2)* system call. Each **shmid** has a segment of memory (referred to as a shared memory segment) and a data structure associated with it. The data structure is referred to as **shmid_ds** and contains the following members:

```
struct ipc_perm shm_perm; /* operation permission struct */
int shm_segsz; /* size of segment */
ushort shm_cpid; /* creator pid */
ushort shm_lpid; /* pid of last operation */
short shm_nattch; /* number of current attaches */
time_t shm_atime; /* last attach time */
time_t shm_dtime; /* last detach time */
time_t shm_ctime; /* last change time */
/* Times measured in secs since */
/* 00:00:00 GMT, Jan. 1, 1970 */
```

Shared memory identifiers can be created using *flok(3C)*.

shm_perm is a **ipc_perm** structure that specifies the permission for a *shmop(2)* or *shmctl(2)* operation (see below). This structure includes the following members:

```
ushort cuid; /* creator user id */
ushort cgid; /* creator group id */
ushort uid; /* user id */
ushort gid; /* group id */
ushort mode; /* r/w permission */
```

shm_segsz specifies the size of the shared memory segment. **shm_cpid** is the process id of the process that created the shared memory identifier. **shm_lpid** is the process id of the last process that performed a *shmop(2)* operation. **shm_nattch** is the number of processes that currently have this segment attached. **shm_atime** is the time of the last *shmat* operation, **shm_dtime** is the time of the last *shmdt* operation, and **shm_ctime** is the time of the last *shmctl(2)* operation that changed one of the members of the above structure.

shared memory operation permissions

In the *shmop(2)* and *shmctl(2)* system call descriptions, the permission required for an operation is indicated for each operation. Whether a particular process has the permission to perform a *shmop(2)* or *shmctl(2)* operation on an object is determined by the object's permission mode bits as follows:

00400	Read by user
00200	Write by user
00060	Read, Write by group
00006	Read, Write by others

Read and Write permissions for a *shmop(2)* or *shmctl(2)* operation on a **shared memory identifier (shmid)** are granted to a process if one or more of the following are true:

- The process's effective user ID is superuser.
- The process's effective user ID matches `shm_perm.[c]uid` in the data structure associated with the `shmid` and the appropriate bit of the "user" portion (0600) of `shm_perm.mode` is set.
- The process's effective user ID does not match `shm_perm.[c]uid` and either the process's effective group ID matches `shm_perm.[c]gid` or one of `shm_perm.[c]gid` is in the process's group access list and the appropriate bit of the "group" portion (060) of `shm_perm.mode` is set.
- The process's effective user ID does not match `shm_perm.[c]uid` and the process's effective group ID does not match `shm_perm.[c]gid` and neither of `shm_perm.[c]gid` is in the process's group access list and the appropriate bit of the "other" portion (06) of `shm_perm.mode` is set.

Otherwise, the corresponding permissions are denied.

shell A user interface to the HP-UX operating system. A shell often functions as both a command interpreter and an interpretive programming language. A shell is automatically invoked for every user who logs in. See *sh(1)* and its related manual entries plus the tutorials supplied with your system for details.

shell program

See **shell script**.

shell script A sequence of shell commands and shell programming language constructs stored in a file and invoked as a user command (program). No compilation is needed prior to execution because the shell recognizes the commands and constructs that make up the shell programming language. A shell script is often called a **shell program** or a **command file**. See the *Shells User Guide*.

shmid See **shared memory identifier**.

signal A software interrupt sent to a process, informing it of special situations or events. Also, the event itself. See *signal(2)*.

single-user state

A condition of the HP-UX operating system in which the system console provides the only communication mechanism between the system and its user. By convention, single-user state is usually specified by *init(1M)* as run-level **S** or **s**. Do not confuse **single-user state**, in which the software is limiting a multiuser system to a single-user communication, with a single-user system, which can never communicate with more than one fixed terminal. See also **multiuser state**.

slash The literal character `/`. A **path name** consisting of a single slash resolves to the **root directory** of the process. See also **path name resolution**.

solidus See **slash**.

source code The fundamental high-level information (program) written in the syntax of a specified computer language. Object (machine-language) code is derived from source code. When dealing with an HP-UX shell command language, **source code** is input to the command language interpreter. The term **shell script** is synonymous with this meaning. When dealing with the C Language, **source code** is input to the *cc(1)* command. **Source code** can also refer to a collection of sources meeting any of the above conditions.

special file A file associated with an I/O device. Often called a **device file**. Special files are read and written the same as **ordinary files**, but requests to read or write result in activation of the associated device. Due to convention and consistency, these files should always reside in the `/dev` directory. See also **file**.

special processes

Processes with certain (small) process IDs are special. On a typical system, the IDs of 0, 1, and 2 are assigned as follows: Process 0 is the scheduler. Process 1 is the initialization process *init*, and is the ancestor of every other process in the system. It is used to control the process structure. On paging systems with virtual memory, process 2 is the paging daemon.

SS/80 See **CS/80**.

standard error

The destination of error and special messages from a program, intended to be used for diagnostic messages. The standard error output is often called **stderr**, and is automatically opened for writing on file descriptor 2 for every command invoked. By default, the user's terminal is the destination of all data written to **stderr**, but it can be redirected elsewhere. Unlike standard input and standard output, which are never used for data transfer in the "wrong" direction, standard error is occasionally read. This is not recommended practice, since I/O redirection is likely to break a program doing this.

standard input

The source of input data for a program. The standard input file is often called **stdin**, and is automatically opened for reading on file descriptor 0 for every command invoked. By default, the user's terminal is the source of all data read from **stdin**, but it can be redirected from another source.

standard output

The destination of output data from a program. The standard output file is often called **stdout**, and is automatically opened for writing on file descriptor 1 for every command invoked. By default, the user's terminal is the destination of all data written to **stdout**, but it can be redirected elsewhere.

stderr See **standard error**.

stdin See **standard input**.

stdout See **standard output**.

stream A term most often used in conjunction with the standard I/O library routines documented in Section 3 of this manual. A stream is simply a file pointer (declared as **FILE *stream**) returned by the *fopen*(3S) library routines. It may or may not have buffering associated with it (by default, buffering is assigned, but this can be modified with *setbuf*(3S)).

sticky bit A single bit in the mode of every file in the file system. If set on a **regular file**, the contents of the file stay permanently in memory instead of being swapped back out to disk when the file has finished executing. Only **superuser** can set the sticky bit on a **regular file**. The sticky bit is read each time the file is executed (via *exec*(2)).

If set on a directory, the files in that directory can be removed or renamed only by the owner of the file, the owner of the directory containing the file, or superuser. See also *chmod*(2), *rename*(2), *rmdir*(2), and *unlink*(2).

subdirectory A directory that is one or more levels lower in the file system hierarchy than a given directory. Sometimes called a **subordinate directory**.

subordinate directory

See **subdirectory**.

Subset 1980 See **CS/80**.

superblock A block on each file system's mass storage medium which describes the file system. The contents of the superblock vary between implementations. Refer to the System Administrator manuals supplied with your system, and the appropriate *fs*(4) entry for details.

superuser The HP-UX system administrator. This user has access to all files, and can perform privileged operations. **superuser** has a **real user ID** and **effective user ID** of 0, and, by convention, the user name of **root**.

superior directory

See **parent directory**.

supplementary group ID

A process has up to **NGROUPS_MAX** supplementary group IDs used in determining file access permissions, in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created.

symbolic link A type of file that indirectly refers to a path name. See *symlink*(4).

- system** The HP-UX operating system. See also **kernel**.
- system asynchronous I/O** A method of performing I/O whereby a process informs a driver or subsystem that it wants to know when data has arrived or when it is possible to perform a write request. The driver or subsystem maintains a set of buffers through which the process performs I/O. See *ioctl(2)*, *read(2)*, *select(2)*, and *write(2)* for more information.
- system call** An HP-UX operating system kernel function available to the user through a high-level language (such as FORTRAN, Pascal, or C). Also called an “intrinsic” or a “system intrinsic.” The available system calls are documented in Section 2 of the *HP-UX Reference Manual*.
- system console** A keyboard and display (or terminal) given a unique status by HP-UX and associated with the special file `/dev/console`. All boot ROM error messages, HP-UX system error messages, and certain system status messages are sent to the system console. Under certain conditions (such as the single-user state), the system console provides the only mechanism for communicating with HP-UX. See the System Administrator manuals and user guides provided with your system for details on configuration and use of the system console.
- system process** A **system process** is a process that runs on behalf of the system. It may have special implementation-defined characteristics.
- terminal** A **character special file** that obeys the specifications of *termio(7)*.
- terminal affiliation** The process by which a process group leader establishes an association between itself and a particular terminal. A terminal becomes affiliated with a process group leader (and subsequently all processes created by the process group leader, see **terminal group**) whenever the process group leader executes (either directly or indirectly) an *open(2)* or *creat(2)* system call to open a terminal. Then, *if* the process which is executing *open(2)* or *creat(2)* is a process group leader, and *if* that process group leader is not yet affiliated with a terminal, and *if* the terminal being opened is not yet affiliated with a process group, the affiliation is established (however, see *open(2)* description of `O_NOCTTY`).
- An affiliated terminal keeps track of its process group affiliation by storing the process group's process group ID in an internal structure.
- Two benefits are realized by terminal affiliation. First, all signals sent from the terminal are sent to all processes in the terminal group. Second, all processes in the terminal group can perform I/O to/from the generic terminal driver `/dev/tty`, which automatically selects the affiliated terminal.
- Terminal affiliation is broken with a terminal group when the process group leader terminates, after which the hangup signal is sent to all processes remaining in the process group. Also, if a process (which is not a process group leader) in the terminal group becomes a process group leader via the *setpgrp(2)* system call, its terminal affiliation is broken.
- See **process group**, **process group leader**, **terminal group**, and *setpgrp(2)*.
- terminal device** See **terminal**.
- text file** A file that contains characters organized into one or more lines. The lines cannot contain NUL characters, and none can exceed `LINE_MAX` bytes in length including the terminating newline character. Although neither the kernel nor the C language implementation distinguishes between text files and binary files (see ANSI C Standard X3-159-19xx), many utilities behave predictably only when operating on text files.
- tty** Originally, an abbreviation for teletypewriter; now, generally, a **terminal**.
- upshifting** The conversion of a lowercase character to its uppercase representation.
- user ID** Each system user is identified by an integer known as a **user ID**, which is in the range of zero to `UID_MAX`, inclusive. Depending on how the user is identified with a process, a **user ID** value is referred to as a **real user ID**, an **effective user ID**, or a **saved user ID**.

UTC	See Epoch .
utility	An executable file, which might contain executable object code (that is, a program), or a list of commands to execute in a given order (that is, a shell script). You can write your own utilities, either as executable programs or shell scripts (which are written in the shell programming language).
volume number	Part of an address used for devices. A number whose meaning is software- and device-dependent, but which is often used to specify a particular volume on a multivolume disk drive. See the System Administrator manuals supplied with your system for details.
whitespace	One or more characters which, when displayed, cause a movement of the cursor or print head, but do not result in the display of any visible graphic. The whitespace characters in the ASCII code set are space, tab, newline, form feed, carriage return, and vertical tab. A particular command or routine might interpret some, but not necessarily all, whitespace characters as delimiting fields, words, or command options.
working directory	Each process has associated with it the concept of a current working directory. For a shell, this appears as the directory in which you currently "reside". This is the directory in which relative path name (i.e., a path name that does not begin with /) searches begin. It is sometimes referred to as the current directory , or the current working directory .
zombie process	The name given to a process which terminates for any reason, but whose parent process has not yet waited for it to terminate (via <i>wait(2)</i>). The process which terminated continues to occupy a slot in the process table until its parent process waits for it. Because it has terminated, however, there is no other space allocated to it either in user or kernel space. It is therefore a relatively harmless occurrence which will rectify itself the next time its parent process waits. The <i>ps(1)</i> command lists zombie processes as defunct .

Index

All Volumes

Index

All Volumes

Index All Volumes

Description	Entry Name(Section)
(CUE), HP Character-Terminal User Environment	cue(1)
(named pipe) special files, make FIFO	mkfifo(1)
(NFS) mount to remote file system, perform Network File System	vhe_u_mnt(1M)
(UNIX) system, call another; terminal emulator	cu(1)
.....	glossary(9)
.....	glossary(9)
.forward file	sendmail(1M)
.netrc - login information for rexec() and ftp	netrc(4)
.o	glossary(9)
.profile file	login(1)
.profile - shell script to set up user's environment at login	profile(4)
.rhosts file	login(1)
.rhosts - security files authorizing access by remote hosts and users on local host	hosts.equiv(4)
.so's from nroff input, eliminate	soelim(1)
/etc files or NIS maps: creating NIS+ tables	nisaddent(1M)
/etc/ioconfig, maintain consistency with kernel data structures	ioinit(1M)
/etc/issue identification file	issue(4)
/etc/lvmpvg - LVM physical volume group information file	lvmpvg(4)
/etc/mnttab, establish mount table	setmnt(1M)
/etc/termcap access routines, emulate	termcap(3X)
/etc/utmp, get login name of user from	getlogin(3C)
/sbin/set_parms special initialization script	hostname(1)
16-bit characters, tools to process	nl_tools_16(3X)
3179G/3192G, 3270, 3777 terminal emulator, IBM	sna(1)
3270, 3777 terminal emulator, IBM 3179G/3192G,	sna(1)
3777 terminal emulator, IBM 3179G/3192G, 3270,	sna(1)
<urses.h> - definition for screen handling and optimisation functions	urses(5)
<math.h> - math functions and constants	math(5)
<regexp.h> - regular expression and pattern matching notation definitions	regexp(5)
<term.h> - terminal capabilities	term(5)
<unctrl.h> - definition for unctrl()	unctrl(5)
<pwd.h> password file format	passwd(4)
a region of window, copy	copywin(3X)
a.out - assembler and link editor output format	a.out(4)
a.out	glossary(9)
a64l() - convert base-64 value to long integer ASCII string	a64l(3C)
abbreviation of function keys, enable/disable	keypad(3X)
abort a per-process timer	rmtimer(3C)
abort() - generate an IOT fault	abort(3C)
abs(), labs() - return integer absolute value	abs(3C)
absolute debugger	adb(1)
absolute path name	glossary(9)
absolute system time, add a specific time interval to the current	get_expiration_time(3T)
absolute value functions	fabs(3M)
absolute value, return integer	abs(3C)
accept - allow LP printer queuing requests	accept(1M)
accept() - accept connection on a socket	accept(2)
access and manage the pathalias database	uupath(1)
access and modification times, set for files	utimes(2)
access control list (ACL) information, get	getacl(2)
access control list (ACL) information, set	setacl(2)
access control list (ACL) structure, convert to string form	acltostr(3C)
access control list (ACL) structure, convert string to	strtoacl(3C)
access control list (ACL), change owner and/or group in	chownacl(3C)
access control list (ACL), copy to another file	cpacl(3C)
access control list; add, modify, or delete entry	setacentry(3C)

Index

All Volumes

Description	Entry Name(Section)
access control lists (ACLs) of files, list	lsacl(1)
access control lists, introduction to	acl(5)
access control lists, view or modify	swacl(1M)
access control lists; add, modify, delete, copy, or summarize	chacl(1)
access device driver, SCSI direct	scsi_disk(7)
access device driver, SCSI sequential	scsi_tape(7)
access exported file system information	exportent(3N)
access functions, NIS+ database	nis_db(3N)
access	glossary(9)
access groups	glossary(9)
access information, network I/O card	lan(7)
access list, get group	getgroups(2)
access list, initialize group	initgroups(3C)
access list, set group	setgroups(2)
access mode	glossary(9)
access or build a binary search tree	tsearch(3C)
access path of physical volume in LVM volume group, change	pvchange(1M)
access permissions mode mask for file-creation, set	umask(1)
access permissions, change file mode	chmod(1)
access permissions, change file mode	chmod(2)
access privileges for group, list	getprivgrp(1)
access privileges, associate group with	setprivgrp(1M)
access protected password database entry	getprpwent(3)
access protections, modify memory mapping	mprotect(2)
access rights to a file, get a user's effective	getaccess(2)
access rights to file(s), list	getaccess(1)
access rights: change on an NIS+ object	nischmod(1)
Access software, Data Communications and Terminal Controller Device File	ddfa(7)
access the terminfo database	tput(1)
access times, set or update file	utime(2)
access to /etc/passwd file, control	lckpwwdf(3C)
access utmp() or wtmp() file	getut(3C)
access utmpx() or wtmp() file	getutx(3C)
access uucp and uux transactions summary log	uucp(1)
access() - determine accessibility of a file	access(2)
access, group, format of privileged values	privgrp(4)
access, modification, and/or change times of a file, update	touch(1)
access, open, or close a directory and associated directory stream	directory(3C)
accessibility of a file, determine	access(2)
account, authentication, session and password management PAM modules for UNIX	pam_unix(5)
account, perform validation procedures for PAM account	pam_acct_mgmt(3)
accounting (taacct) file, print any total	acctsh(1M)
accounting data and command usage summary, accumulate	acctsh(1M)
accounting data of VxFS file system, disk usage by user ID	vxdiskusg(1M)
accounting data, disk usage by user ID	diskusg(1M)
accounting files, process, convert to ASCII text format	acctprc(1M)
accounting files, process, summarize by user ID and name	acctprc(1M)
accounting files, search and print	acctcom(1M)
accounting files, total, merge or add	acctmerg(1M)
accounting records, manipulate	fwtmp(1M)
accounting records, per-process, command summary from	acctcms(1M)
accounting summary files, create periodic	acctsh(1M)
accounting: acctcms - command summary from per-process accounting records	acctcms(1M)
accounting: acctcon1 - convert login/logoff records to per-session accounting records	acctcon(1M)
accounting: acctcon2 - convert per-session records to total accounting records	acctcon(1M)
accounting: acctdisk - create disk usage accounting records	acct(1M)
accounting: acctdusg - compute disk usage by login name	acct(1M)
accounting: acctmerg - merge or add total accounting files	acctmerg(1M)
accounting: accton - define kernel process accounting output file or disable accounting	acct(1M)
accounting: acctprc1 - convert process accounting files to ASCII text format	acctprc(1M)
accounting: acctprc2 - summarize process accounting files created by acctprc1	acctprc(1M)

Description	Entry Name(Section)
accounting: acctwtmpt - write utmp record and reason for writing	acct(1M)
accounting: chargefee - charge fee to user based on system usage	acctsh(1M)
accounting: check size of process accounting file	acctsh(1M)
accounting: ckpacct - check size of process accounting file	acctsh(1M)
accounting: daily accounting shell procedure	runacct(1M)
accounting: dodisk - perform disk accounting	acctsh(1M)
accounting: enable or disable process accounting	acct(2)
accounting: lastlogin - show last login date for each user	acctsh(1M)
accounting: monacct - create periodic accounting summary files	acctsh(1M)
accounting: nulladm - create empty file owned by adm with mode 664	acctsh(1M)
accounting: per-process accounting file format	acct(4)
accounting: perform disk accounting	acctsh(1M)
accounting: prctmpt - print session record file created by acctcon1	acctsh(1M)
accounting: prdaily - print daily accounting report	acctsh(1M)
accounting: prtacct - print any total accounting (tacct) file	acctsh(1M)
accounting: runacct - accumulate accounting data and command usage summary	acctsh(1M)
accounting: shell procedures for system accounting	acctsh(1M)
accounting: shutacct - turn off accounting for system shutdown	acctsh(1M)
accounting: startup - start accounting process at system startup	acctsh(1M)
accounting: turnacct - turn on or turn off process accounting	acctsh(1M)
accounting: user accounting file entry format	utmp(4)
acct - per-process accounting file format	acct(4)
acct() - enable or disable process accounting	acct(2)
acctcms - command summary from per-process accounting records	acctcms(1M)
acctcom - search and print process accounting files	acctcom(1M)
acctcon1 - convert login/logoff records to per-session accounting records	acctcon(1M)
acctcon1 - print session record file created by	acctsh(1M)
acctcon2 - convert per-session records to total accounting records	acctcon(1M)
acctdisk - create disk usage accounting records	acct(1M)
acctdusg - compute disk usage by login name	acct(1M)
acctmerg - merge or add total accounting files	acctmerg(1M)
accton - define kernel process accounting output file or disable accounting	acct(1M)
acctprc - convert process accounting files to ASCII text format	acctprc(1M)
acctprc1 - convert process accounting files to ASCII text format	acctprc(1M)
acctprc2 - summarize process accounting files created by acctprc1	acctprc(1M)
acctwtmpt - write utmp record and reason for writing	acct(1M)
ACL, view or modify	swacl(1M)
aclentrystart() - convert string to access control list (ACL) structure	strtoacl(3C)
ACLs	see access control lists
acltostr() - convert access control list (ACL) structure to string form	acltostr(3C)
acos() - arccosine function	acos(3M)
acosd() - arccosine function (degrees)	acosd(3M)
acosdf() - arccosine function (float, degrees)	acosd(3M)
acosf() - arccosine function (float)	acos(3M)
acosh() - inverse hyperbolic cosine function	acosh(3M)
active controllers on HP-IB, change	hplib_pass_ctl(3I)
active processes, kill (terminate) all	killall(1M)
activity on specified HP-IB bus, stop	hplib_abort(3I)
activity reporter, system	sar(1M)
activity, print current SCCS file editing	sact(1)
activity, system, daily report package	sa1(1M)
adb - absolute debugger	adb(1)
add a complex character and rendition to a window	add_wch(3X)
add a new group to the system	groupadd(1M)
add a printer for use with tsm	tsm.lpadmin(1M)
add a single-byte character and rendition to a window and advance the cursor	addch(3X)
add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
add a user login on the system	useradd(1M)
add a wide-character string to a window and advance the cursor	addnwstr(3X)
add an array of complex characters and renditions to a window	add_wchnstr(3X)
add length limited string of single-byte characters and renditions to a window	addchnstr(3X)

Index All Volumes

Description	Entry Name(Section)
add line number in front of each line in a file	nl(1)
add new commands to system	install(1M)
add or merge total accounting files	acctmerge(1M)
add physical volumes to extend an LVM volume group	vgextend(1M)
add string of single-byte characters and renditions to a window	addchstr(3X)
add swap device for interleaved paging and swapping	swapon(2)
add value to environment	putenv(3C)
add, delete, update a kernel module	kminstall(1M)
add, modify, delete, copy, or summarize file access control lists (ACLs)	chacl(1)
add, modify, or delete access control list entry	setaclentry(3C)
addch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
addchnstr() - add length limited string of single-byte characters and renditions to a window ...	addchnstr(3X)
addchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
addexportent() - access exported file system information	exportent(3N)
additional cursor and window coordinates, get	getbegyx(3X)
additional severities, define	addsev(3C)
addmntent() - add entry to open file system description file	getmntent(3X)
addnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
addnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
address	glossary(9)
address manipulation routines, Internet	inet(3N)
address mapping, physical memory	iomap(7)
address of connected peer, get	getpeername(2)
address resolution display and control	arp(1M)
address resolution protocol	arp(7P)
address router, electronic	pathalias(1)
address space	glossary(9)
address string conversion routines, network station	net_aton(3C)
address to a socket, bind	bind(2)
address, get socket	getsockname(2)
addresses - first locations beyond allocated program regions	end(3C)
addsev() - define additional severities	addsev(3C)
addstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
addwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
add_wch() - add a complex character and rendition to a window	add_wch(3X)
add_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
add_wchstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
adjtime() - correct the time to synchronize the system clock	adjtime(2)
adjust - simple text formatter	adjust(1)
adjustment table, time zone, for date and ctime	tztab(4)
admin - create and administer SCCS files	admin(1)
administer and create SCCS files	admin(1)
administer LicensePower/iFOR licensing	i4admin(1M)
administer NIS+ groups	nisgrpadm(1)
administer NIS+ tables	nistbladm(1)
administration command for file system	fsadm(1M)
administration command for HFS file system	fsadm_hfs(1M)
administration commands, file system configuration and binary files	fs_wrapper(5)
administration file owned by adm with mode 664, create empty	acctsh(1M)
administration functions, NIS+ log	nis_ping(3N)
administration manager, system, menu-driven	sam(1M)
administration, kernel module	kmadmin(1M)
administration, local area network	lanadmin(1M)
administration: STREAMS Administrative Driver	sad(7)
administrator, system	passwd(1)
ADVANCE() - advance pointer to next 8- or 16-bit character	nl_tools_16(3X)
advance() - regular expression substring comparison routines	regexp(3X)
AdvanceLink server, Basic Serial and HP	pcserver(1M)
advances a job to the top of a queue	pdpromote(1)

Description	Entry Name(Section)
advise system of process's expected paging behavior	madvise(2)
affiliation	glossary(9)
affirmative responses, repetitively	yes(1)
aio() - POSIX asynchronous I/O facility	aio(5)
aio_cancel() - cancel asynchronous I/O operation	aio_cancel(2)
aio_error() - return error status of asynchronous I/O operation	aio_error(2)
aio_fsync() - bring asynchronous I/O operations to synchronized state	aio_fsync(2)
aio_read() - start asynchronous read operation	aio_read(2)
aio_return() - return asynchronous I/O operation status	aio_return(2)
aio_suspend() - suspend for asynchronous I/O completion	aio_suspend(2)
aio_suspend() - wait for asynchronous I/O completion	aio_suspend(2)
aio_write() - start asynchronous write operation	aio_write(2)
alarm clock, set a process's	alarm(2)
alarm() - set a process's alarm clock	alarm(2)
alias database	elm(1)
alias file, create sendmail	sendmail(1M)
alias - substitute command and/or file name	sh-posix(1)
alias - substitute command and/or filename	csh(1)
alias - substitute command and/or filename	ksh(1)
alias text file	elm(1)
alias: install new elm aliases for user or system	newaliases(1)
aliases - aliases file for sendmail	aliases(5)
aliases and paths, locate a program file including	which(1)
aliases file for sendmail	aliases(5)
aliases file, for mail, rebuild the database	newaliases(1M)
aliases, elm user and system, verify and display	elmaliases(1)
aliases, print system-wide sendmail	praliases(1)
all users over a network, write to	rwall(1M)
alloc - show dynamic memory usage	csh(1)
alloca() - allocate space from the stack	malloc(3C)
allocate a per-process timer	mktimer(3C)
allocate data and stack space then lock process into memory	datalock(3C)
allocate library structure for transport function argument structures (X/OPEN TLI-XTI)	t_alloc(3)
allocate reserved space for a disk storage file	prealloc(1)
allocated physical extents, move from one LVM physical volume to other physical volumes	pvmove(1M)
allocated program regions, first locations beyond	end(3C)
allocation space of object files, print section sizes and	size(1)
allocation, change data segment space	brk(2)
allocator for main memory	malloc(3C)
allow interface to enable SRQ line on HP-IB	hpib_rqst_srvce(3I)
allow signals to interrupt functions	siginterrupt(2)
allowed login shells, list of	shells(4)
alphasort() - sort a directory pointer array	scandir(3C)
alter contents of and copy a (tape) file	dd(1)
alter priority of running processes	renice(1)
alter selected characters	tr(1)
alternate stack context, set/get signal alternate stack context	sigaltstack(2)
an array of complex characters and renditions to a window, add	add_wchnstr(3X)
analysis information, print LP spooler performance	lpana(1M)
ancillary agent for inbound zone transfer	named-xfer(1M)
anonymous memory region, initialize semaphore in mapped file or	msem_init(2)
anonymous region, remove semaphore in mapped file or	msem_remove(2)
another system over LAN, log in on	vt(1)
answer - phone message transcription system	answer(1)
APIs, PAM Service Module APIs	pam_sm(3)
append characters to obtain text line of specified length	newform(1)
application, explicit locking of streams within a multi-thread	flockfile(3S)
application, header file for future applications	portal(5)
applying a diff file to an original	patch(1)
appropriate privileges	glossary(9)
appropriate privileges, associate group with certain	setprivgrp(1M)

Index

All Volumes

Description	Entry Name(Section)
ar - archive and library maintainer for portable archives	ar(1)
ar - common archive file format	ar(4)
arbitrary-precision arithmetic language	bc(1)
arccosine functions (degrees)	acosd(3M)
arccosine functions	acos(3M)
archive and library maintainer for portable archives	ar(1)
archive exchange, portable	pax(1)
archive file format, common	ar(4)
archive format, cpio	cpio(4)
archive format, tar tape	tar(4)
archive	glossary(9)
archive member access for ELF files	elf_rand(3E)
archive member header for ELF files, retrieve	elf_getarhdr(3E)
archive package, make an	shar(1)
archive symbol table for ELF files, retrieve	elf_getarsym(3E)
archive symbol table, regenerate	ranlib(1)
archive the file system	backup(1M)
archiver for tape files	tar(1)
archiver, tape file	tar(1)
archives, copy file archives in and out	cpio(1)
arcsine functions (degrees)	asind(3M)
arcsine functions	asin(3M)
arctangent functions (degrees)	atand(3M)
arctangent functions	atan(3M)
arctangent-and-quadrant functions (degrees)	atan2d(3M)
arctangent-and-quadrant functions	atan2(3M)
argument list(s), large, construct and execute command	xargs(1)
argument list, print formatted output of a varargs	vprintf(3S)
argument lists, variable, macros for handling	stdarg(5)
argument lists, variable, macros for handling	varargs(5)
argument vector, get option letter from	getopt(3C)
argument, varargs, formatted input conversion to a	vscanf(3S)
arguments, echo (print)	echo(1)
arguments, evaluate as an expression	expr(1)
arguments, print formatted	printf(1)
arguments, too many (error); construct argument list(s) and execute command	xargs(1)
arithmetic desk calculator	dc(1)
arithmetic language, arbitrary-precision	bc(1)
arm a per-process timer, relatively	reltimer(3C)
arp - address resolution display and control	arp(1M)
arp - address resolution protocol	arp(7P)
array element, convert floating-point number to string or string	ecvt(3C)
array of complex characters and renditions, input from a window	in_wchnstr(3X)
array of single-byte characters and renditions, input from a window	inchnstr(3X)
array of wide characters and function key codes, get from a terminal	getn_wstr(3X)
array, sort a directory pointer	scandir(3C)
arrayinfo -describe disk array characteristics	arrayinfo(1M)
arrayscan - search system for disk arrays	arrayscan(1M)
arraytab - disk array configuration table	arraytab(4)
as - assembler (Precision Architecture)	as(1)
ASA carriage control characters, interpret	asa(1)
asa - interpret ASA carriage control characters	asa(1)
ASCII format, dump iconv translation tables to	dmpxlt(1)
ASCII	glossary(9)
ascii - map of ASCII character set	ascii(5)
ASCII string, convert between long integer and base-64	a64l(3C)
ASCII string, convert long integer to	ltostr(3C)
ASCII, 7-bit, translate characters to	conv(3C)
ASCII, convert binary file to, for transmission by mailer	uuencode(1)
asctime() , asctime_r() - convert tm structure date and time to string	ctime(3C)
asecure - control access to Audio	asecure(1M)

Description	Entry Name(Section)
aserver	aserver(1M)
aserver - Audio	aserver(1M)
asin() - arcsine function	asin(3M)
asind() - arcsine function (degrees)	asind(3M)
asindf() - arcsine function (float, degrees)	asind(3M)
asinf() - arcsine function (float)	asin(3M)
asinh() - inverse hyperbolic sine function	asinh(3M)
ask for user response for SD-UX	swask(1M)
assembler (Precision Architecture)	as(1)
assembler and link editor output format	a.out(4)
assembler debugger	adb(1)
assert() - verify program assertion	assert(3X)
assertion, verify program	assert(3X)
assign buffering to a stream file	setbuf(3S)
assignment database entry, manipulate device	getdvagnt(3)
asynchronous I/O, error status	aio_error(2)
asynchronous I/O, initiate list of operations	lio_listio(2)
asynchronous I/O, POSIX	aio(5)
asynchronous I/O, start write	aio_write(2)
asynchronous I/O, status, return	aio_return(2)
asynchronous I/O, suspend for completion	aio_suspend(2)
asynchronous I/O, synchronize	aio_fsync(2)
asynchronous I/O, wait for completion	aio_suspend(2)
asynchronous serial modem line control	modem(7)
asynchronous, cancel I/O	aio_cancel(2)
asynchronous, start read	aio_read(2)
at, batch, and crontab queue description file	queuedefs(4)
at, prototype job file for	proto(4)
at - execute commands at a later time	at(1)
atan() - arctangent function	atan(3M)
atan2() - arctangent-and-quadrant function	atan2(3M)
atan2d() - arctangent-and-quadrant function (degrees)	atan2d(3M)
atan2df() - arctangent-and-quadrant function (float, degrees)	atan2d(3M)
atan2f() - arctangent-and-quadrant function (float)	atan2(3M)
atand() - arctangent function (degrees)	atand(3M)
atandf() - arctangent function (float, degrees)	atand(3M)
atanf() - arctangent function (float)	atan(3M)
atanh() - inverse hyperbolic tangent function	atanh(3M)
atexit() - register a function to be called at program termination	atexit(2)
atof() - convert string to double-precision number	strtod(3C)
atoi() - convert string to long integer	strtol(3C)
atol() - convert string to long integer	strtol(3C)
atomically release blocked signals and wait for interrupt	sigpause(2)
attach a STREAMS file descriptor to an object in the file system name space	fattach(3C)
attach shared memory to data segment	shmop(2)
attempt to lock a read-write lock for writing	pthread_rwlock_wrlock(3T)
attribute object, initialize or destroy thread	pthread_attr_dinit(3T)
attribute, for window, control functions	attr_get(3X)
attributes associated with a message queue, get	mq_getattr(2)
attributes - describe audio file	attributes(1)
attributes for group, get special	getprivgrp(1)
attributes of a DOS file, change	doschmod(1)
attributes, change program's internal	chatr(1)
attributes, change RCS file	racs(1)
attributes, objects, and storage formats for SD	sd(4)
attributes, set and get for pthread	pthread_attr_getdetachstate(3T)
attributes, window, set and clear	standend(3X)
attroff() - restricted window attribute control functions	attroff(3X)
attron() - restricted window attribute control functions	attroff(3X)
attrset() - restricted window attribute control functions	attroff(3X)
attr_get() - window attribute control functions	attr_get(3X)

Index

All Volumes

Description	Entry Name(Section)
<code>attr_off()</code> - window attribute control functions	<code>attr_get</code> (3X)
<code>attr_on()</code> - window attribute control functions	<code>attr_get</code> (3X)
<code>attr_set()</code> - window attribute control functions	<code>attr_get</code> (3X)
<code>audctl(2)</code> - HP-UX Auditing System described	<code>audit</code> (5)
<code>audctl1()</code> - start or halt auditing system; set or get audit files	<code>audctl</code> (2)
<code>audevent</code> (1M) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audevent</code> - change or display event or system call audit status	<code>audevent</code> (1M)
<code>audeventstab</code> - define and describe audit system events	<code>audeventstab</code> (4)
audible signal	<code>beep</code> (3X)
audio - audio tools available through HP VUE	<code>audio</code> (5)
Audio	<code>Audio</code> (1)
audio control panel	see <code>audio</code> (5)
audio editor,	see <code>audio</code> (5)
audio file and data formats	see <code>audio</code> (5)
audio file, convert	<code>convert</code> (1)
audio file, describe	<code>attributes</code> (1)
audio file, play	<code>send_sound</code> (1)
audio library	see <code>audio</code> (5)
audio security	see <code>audio</code> (5)
audio setup	see <code>audio</code> (5)
audio, control access to	<code>asecure</code> (1M)
<code>audisp</code> (1M) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audisp</code> - display audit information as requested by parameters	<code>audisp</code> (1M)
<code>audit</code> (5) - HP-UX Auditing System described	<code>audit</code> (5)
audit - change or display event or system call audit status	<code>audevent</code> (1M)
audit files, set or get	<code>audctl</code> (2)
audit ID (<code>aid()</code>) for current process, get	<code>getaudit</code> (2)
audit ID (<code>aid()</code>), set for current process	<code>setaudit</code> (2)
audit information, display as requested by parameters	<code>audisp</code> (1M)
audit process flag for calling process, get	<code>getaudproc</code> (2)
audit record, write for self-auditing process	<code>audwrite</code> (2)
audit system events, define and describe	<code>audeventstab</code> (4)
audit-overflow monitor daemon	<code>audomon</code> (1M)
audit: audit-overflow monitor daemon	<code>audomon</code> (1M)
audit: file format and other information for auditing	<code>audit</code> (4)
audit: get events and system calls currently being audited	<code>getevent</code> (2)
audit: select users to audit	<code>audusr</code> (1M)
audit: set current events and system calls to be audited	<code>setevent</code> (2)
audit: set or clear auditing on calling process	<code>setaudproc</code> (2)
audit: set or display audit file information	<code>audsys</code> (1M)
audit: set or get audit files	<code>audctl</code> (2)
audit: start or halt auditing system	<code>audctl</code> (2)
audit: start or halt auditing system	<code>audsys</code> (1M)
auditing system	see <code>audit</code>
auditing system, start or halt	<code>audctl</code> (2)
auditing, set or clear on calling process	<code>setaudproc</code> (2)
auditing, suspend or resume on current process	<code>audswitch</code> (2)
<code>audomon</code> - audit-overflow monitor daemon	<code>audomon</code> (1M)
<code>audswitch</code> (2) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audswitch()</code> - suspend or resume auditing on current process	<code>audswitch</code> (2)
<code>audsys</code> (1M) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audsys</code> - start or halt the auditing system and set or display audit file information	<code>audsys</code> (1M)
<code>audusr</code> (1M) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audusr</code> - select users to audit	<code>audusr</code> (1M)
<code>audwrite</code> (2) - HP-UX Auditing System described	<code>audit</code> (5)
<code>audwrite()</code> - write audit record for self-auditing process	<code>audwrite</code> (2)
<code>authcap</code> - security databases	<code>authcap</code> (4)
<code>authck</code> - check internal consistency of Authentication database	<code>authck</code> (1M)
<code>authdes_create()</code> - obsolete library routines for RPC	<code>rpc_soc</code> (3N)
<code>authdes_seccreate()</code> - library routines for secure remote procedure calls	<code>secure_rpc</code> (3N)
authentication and print request server, PC-NFS	<code>pcnfsd</code> (1M)

Description	Entry Name(Section)
Authentication database, check internal consistency of	authck(1M)
authentication file format, for ppp	ppp.Auth(4)
authentication information routines for PAM	pam_set_item(3)
authentication module, configuration file for pluggable authentication module	pam.conf(4)
authentication module, pluggable	pam(3)
authentication service, modify and delete user credentials for an authentication service	pam_setcred(3)
authentication transaction routines for PAM	pam_start(3)
authentication within the PAM framework, perform	pam_authenticate(3)
authentication, account, session and password management PAM modules for UNIX	pam_unix(5)
authnone_create() - library routines for client side remote procedure call authentication ..	rpc_clnt_auth(3N)
authsys_create() - library routines for client side remote procedure call authentication ..	rpc_clnt_auth(3N)
authsys_default() - library routines for client side remote procedure call authentication ..	rpc_clnt_auth(3N)
authunix_create() - obsolete library routines for RPC	rpc_soc(3N)
authunix_create_default() - obsolete library routines for RPC	rpc_soc(3N)
auth_destroy() - library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
autoboot sequence	pdc(1M)
automatically mount NFS file systems	automount(1M)
automount - automatically mount NFS file systems	automount(1M)
autopush - manage system database of automatically pushed STREAMS modules	autopush(1M)
autox0 - SCSI media changer device drivers	autochanger(7)
auto_parms - Initial system configuration/DHCP support script	auto_parms(1M)
availability (LVM), set volume group	vgchange(1M)
available DOS disk clusters, report number of	dosdf(1)
available, login when VHE home machine is not	vhe_altlog(1M)
awk - text pattern scanning and processing language	awk(1)
back into input stream, push character	ungetc(3S)
back into input stream, push wide character	ungetwc(3C)
background batch execution	at(1)
background character and rendition using a complex character	bkgrnd(3X)
background character and rendition using a single-byte character	bkgd(3X)
background process group	glossary(9)
background processes to complete, wait for	wait(1)
backspaces and reverse line-feeds, remove from text	col(1)
backup - backup or archive the file system	backup(1M)
backup file, create or update LVM volume group configuration	vgcfbackup(1M)
backup	glossary(9)
backup, incremental file system dump	dump(1M)
backup, incremental file system dump over network	dump(1M)
banner - make posters in large letters	banner(1)
base offset for an object file, get	elf_getbase(3E)
base two logarithm functions	log2(3M)
base-2 exponential function	exp2(3M)
base-64 ASCII string, convert long integer to	a64l(3C)
basename, dirname - extract portions of path names	basename(1)
basename() - return final component of path name	basename(3C)
basic integer data types	inttypes(5)
Basic Serial and HP AdvanceLink server	pcserver(1M)
batch, at, and crontab queue description file	queuedefs(4)
batch - execute commands immediately	at(1)
batch mail interface	fastmail(1)
baud rate, get terminal	baudrate(3X)
baud rate, tty, set or get	cfspeed(3C)
baudrate() - get terminal baud rate	baudrate(3X)
bc - arbitrary-precision arithmetic language	bc(1)
bcmp() - BSD memory compare	memory(3C)
bcopy() - BSD memory copy	memory(3C)
bdf - report number of free disk blocks (Berkeley version)	bdf(1M)
bdiff - big diff	bdiff(1)
beep() - audible signal	beep(3X)
beginning of file, list first few lines at	head(1)
behavior, advise system of process's expected paging	madvise(2)

Index

All Volumes

Description	Entry Name(Section)
Bessel functions of the first kind	j0(3M)
Bessel functions of the second kind	y0(3M)
bfs - big file scanner	bfs(1)
bgets() - read stream up to next delimiter	bgets(3G)
big diff	bdiff(1)
big file scanner	bfs(1)
bill fee to user based on system usage	acctsh(1M)
binary directories, install object files in	cpset(1M)
binary file, convert to ASCII for transmission by mailer	uuencode(1)
binary files used by file system administration commands	fs_wrapper(5)
binary files, format tracing and logging	netfmt(1M)
binary input/output to a stream file, buffered	fread(3S)
binary or object file, find the printable strings in an	strings(1)
binary program files for given name, find location of	whereis(1)
binary search routine for sorted tables	bsearch(3C)
binary search tree, manage a	tsearch(3C)
bind a socket to a privileged IP port	bindresvport(3N)
bind address to transport endpoint (X/OPEN TLI-XTI)	t_bind(3)
bind services, library routines for RPC	rpcbind(3N)
bind threads to processors	pthread_processor_bind_np(3T)
bind to a particular Network Information Service server	ypset(1M)
bind() - bind an address to a socket	bind(2)
binder processes, Network Information Service (NIS)	ypserv(1M)
bindresvport() - bind a socket to a privileged IP port	bindresvport(3N)
biod - NFS block I/O daemon	nfsd(1M)
bit bucket	null(7)
bkgd() - set or get background character and rendition using a single-byte character	bkgd(3X)
bkgrnd() - set or get background character and rendition using omplex character	bkgrnd(3X)
blank lines, reduce multiple adjacent to single blank line	ssp(1)
blank lines, remove all from file	rnnl(1)
blclose() - terminal block-mode library interface	blmode(3C)
blget() - terminal block-mode library interface	blmode(3C)
blmode - terminal block mode interface	blmode(7)
blmode() - terminal block-mode library interface	blmode(3C)
block count and checksum of a file, print	sum(1)
block count and checksum of a file, print	sum(1)
block	glossary(9)
block mode terminal interface	blmode(7)
block signals	sigblock(2)
block size, dump file system	dumpfs(1M)
block special file	glossary(9)
block, enable or disable during read	nodelay(3X)
block-mode terminal I/O library interface	blmode(3C)
blocked signals, examine and change	sigprocmask(2)
blocked signals, release and atomically wait for interrupt	sigpause(2)
blocking on input, control	notimeout(3X)
blocking status of a message queue associated with a descriptor, set	mq_setattr(2)
blocks, report number of free disk (Berkeley version)	bdf(1M)
blopen() - terminal block-mode library interface	blmode(3C)
blread() - terminal block-mode library interface	blmode(3C)
blset() - terminal block-mode library interface	blmode(3C)
boot area	glossary(9)
boot device configuration table	bootconf(4)
boot	glossary(9)
boot programs; install, update, or remove from a disk device	mkboot(1M)
Boot Protocol server, Internet	bootpd(1M)
boot ROM	glossary(9)
boot - run bootstrap process	boot(1M)
boot server, remote (diskless)	rbootd(1M)
boot the system	reboot(2)
boot, loadable modules for running kernel during boot	loadmods(4)

Description	Entry Name(Section)
boot-up	glossary(9)
bootable, build a bootable HP-UX kernel or kernel modules	mk_kernel(1M)
bootconf - boot device configuration table	bootconf(4)
BOOTP server, send BOOTREQUEST to	bootquery(1M)
bootpd - Internet Boot Protocol server	bootpd(1M)
bootpd, command line tools for DHCP elements of	dhcptools(1M)
bootpquery - send BOOTREQUEST to BOOTP server	bootpquery(1M)
bootptab entry, get or put	getbootpt(3X)
BOOTREQUEST, send to BOOTP server	bootpquery(1M)
bootstrap and installation utility, HP-UX	hpux(1M)
bootstrap process, run	boot(1M)
border() - draw borders from single-byte characters and renditions	border(3X)
borders, draw from complex characters and renditions	border_set(3X)
borders, draw from complex characters and renditions	box_set(3X)
borders, draw from single-byte characters and renditions	border(3X)
borders, draw from single-byte characters and renditions	box(3X)
border_set() - draw borders from complex characters and renditions	border_set(3X)
Bourne shell	sh-bourne(1)
box() - draw borders from single-byte characters and renditions	box(3X)
box_set() - draw borders from complex characters and renditions	box_set(3X)
break a file into multiple <i>n</i> -line pieces	split(1)
break - exit from enclosing for/next loop	csh(1)
break - exit from enclosing for/next loop	ksh(1)
break - exit from enclosing for/next loop	sh-bourne(1)
break - exit from enclosing for/next loop	sh-posix(1)
break value and file size limits, get or set	ulimit(2)
breaksw - break from switch and resume after endsw	csh(1)
brk(), sbrk() - change data segment space allocation	brk(2)
broadcast message simultaneously to all users	wall(1M)
bs - a compiler/interpreter for modest-sized programs	bs(1)
BSD-4.2-compatible kill(), sigvec(), and signal() system calls	bsdproc(3C)
BSD-4.2-compatible kill(), sigvec(), and signal() system calls	killpg(2)
bsd_signal() - simplified signal facilities	bsd_signal(3C)
bsearch() - binary search routine for sorted tables	bsearch(3C)
bss (uninitialized data) allocation space of object files, print section sizes and	size(1)
btmp file	login(1)
btmp(), utmp(), wtmp() - utmp, wtmp, btmp user accounting file entry format	utmp(4)
buffer, flush with or without closing stream	fclose(3S)
buffer, split into fields	bufsplit(3G)
buffered binary input/output to a stream file	fread(3S)
buffered input/output standard stream file package	stdio(3S)
buffering, assign to a stream file	setbuf(3S)
buffers, flush to disk	sync(2)
buffers, flush unwritten system buffers to disk	sync(1M)
buffers, periodically flush unwritten system buffers to disk	syncer(1M)
buffers, use to perform I/O with an HP-IB channel	hpib_io(3I)
bufsplit() - split buffer into fields	bufsplit(3G)
build a bootable HP-UX kernel or kernel modules	mk_kernel(1M)
build a makefile	mkmf(1)
build an HP-UX system	config(1M)
build and install Network Information Service databases	ypinit(1M)
build or access a binary search tree	tsearch(3C)
bundled C compiler	cc_bundled(1)
bus address	glossary(9)
bus	see HP-IB
bus, stop activity on specified HP-IB	hpib_abort(3I)
byte order, network and host, convert values between	byteorder(3N)
byte, compare memory contents with specified	memory(3C)
byte, find location of in memory	memory(3C)
byte, set contents of memory area to specified	memory(3C)
bytes in a file, count	wc(1)

Index

All Volumes

Description	Entry Name(Section)
bytes over HP-IB, send command	hpib_send_cmnd(3I)
bytes, swap	swab(3C)
byte_status() , BYTE_STATUS() - test for valid 1- or 2-byte character	nl_tools_16(3X)
bzero() - BSD memory clear	memory(3C)
C and Pascal execution startup routines	crt0(3)
C header files, generate	rpcgen(1)
C language preprocessor	cpp(1)
C language, process include and conditional instructions	cpp(1)
C macro processor	m4(1)
C programs, extract strings from C programs to implement shared strings	xstr(1)
C source, extract error messages from into a file	mkstr(1)
C-like syntax, a shell (command interpreter) with	csh(1)
cache file, share, NIS+ utility to print out the contents of	nisshowcache(1M)
cache location information about NIS+ servers	nis_cachemgr(1M)
caching and hashing daemon, password and group	pwgrd(1M)
caching and hashing statistics, password and group	pwgr_stat(1M)
caching: controlling on HP SCSI disk arrays	dcc(1M)
cal - print calendar	cal(1)
calculate default disk section sizes	disksecn(1M)
calculator, desk	dc(1)
calendar - reminder service	calendar(1)
calendar, print	cal(1)
call an initialization routine only once, threads	pthread_once(3T)
call another (UNIX) system; terminal emulator	cu(1)
call graph execution profile data, display	gprof(1)
call terminal- spawn getty to remote terminal	ct(1)
call, data returned by stat/fstat/lstat	stat(5)
calling process, get audit process flag for	getaudproc(2)
calling process, set or clear auditing on	setaudproc(2)
calling process, suspend	napms(3X)
calloc() - allocate memory for array	malloc(3C)
callrpc() - obsolete library routines for RPC	rpc_soc(3N)
calls, system, BSD-4.2-compatible kill() , sigvec() , and signal()	bsdproc(3C)
calls, system, BSD-4.2-compatible kill() , sigvec() , and signal()	killpg(2)
cancel a notification request with a message queue	mq_notify(2)
cancel a per-process timer	rmtimer(3C)
cancel asynchronous I/O	aio_cancel(2)
cancel - cancel requests on an LP printer	lp(1)
cancel execution of a thread	pthread_cancel(3T)
cancel LP requests from spooling queue on remote system	rcancel(1M)
cancel requests on an LP printer	lp(1)
cancellability state and type, set and retrieve the current thread's	pthread_setcancelstate(3T)
cancellation cleanup handler, register or remove thread	pthread_cleanup_pop(3T)
cancellation requests, process any pending	pthread_testcancel(3T)
can_change_color() - color manipulation functions	can_change_color(3X)
capabilities, retrieve from the terminfo database	tigetflag(3X)
capabilities, terminal, get from terminfo database	tput(1)
captoinfo - convert a termcap description into a terminfo description	captoinfo(1M)
card access information, network I/O	lan(7)
carriage control characters, ASA, interpret	asa(1)
cartridge tape device access drivers	ct(7)
Cartridge Tape I/O Utility, CS/80	tcio(1)
cartridge tape, hard disk, or flexible disk media, initialize	mediainit(1)
cartridge, partition DDS tape	mediainit(1)
case - execute <i>list</i> associated with <i>pattern</i> that matches <i>word</i>	ksh(1)
case - execute <i>list</i> associated with <i>pattern</i> that matches <i>word</i>	sh-posix(1)
case - label in a switch statement	csh(1)
cat a VxFS file system	fs_cat_vxfs(1M)
cat after uncompressing Huffman coded files (see <i>pack(1)</i>)	compact(1)
cat - concatenate, copy, and print files	cat(1)
cat files for on-line manual pages, create	catman(1M)

Description	Entry Name(Section)
catalog file, generate a formatted message	gencat(1)
catalog file, message, create for modification	findmsg(1)
catalog for reading, close or open NLS message	catopen(3C)
catalog, set the default message	setcat(3)
catalogs, message, find strings for inclusion in	findstr(1)
catclose() - close NLS message catalog for reading	catopen(3C)
catgets(3C) , insert calls to based on <i>findstr(1)</i> output	insertmsg(1)
catgets() - get an NLS program message	catgets(3C)
catman - create the cat files for the manual	catman(1M)
catopen() - open NLS message catalog for reading	catopen(3C)
cause the calling thread to terminate	pthread_exit(3T)
cbreak() - input mode control functions	cbreak(3X)
cbrt() , cbrtf() - cube root functions	cbrt(3M)
cbrtf() , cbrt() - cube root functions	cbrt(3M)
ccat - uncompact and cat files using Huffman code (see <i>pack(1)</i>)	compact(1)
cchar_t , get a wide character string and rendition from	getcchar(3X)
cchar_t , set from a wide character string and rendition	setcchar(3X)
cc_bundled - bundled C compiler	cc_bundled(1)
cd - change working directory	cd(1)
cd - change working directory	cd(1)
cd - change working directory	cd(1)
cd - change working directory	cd(1)
cd - change working directory	cd(1)
CD-ROM file systems, mount and unmount	pfs_mount(1M)
CD-ROM: background information	cdrom(4)
CD-ROM: format of a CDFS cnode	cdnode(4)
cdc - change the delta commentary of an SCCS delta	cdc(1)
CDFS cnode, format of a	cdnode(4)
CDFS file system disk blocks, report number of free	df_hfs(1M)
CDFS file systems, mount and unmount	mount_cdfs(1M)
cdnode - format of a CDFS cnode	cdnode(4)
cdrom - CD-ROM background information	cdrom(4)
ceil() - ceiling function	ceil(3M)
ceiling function	ceil(3M)
cent - Centronics-compatible interface	cent(7)
Centronics-compatible interface	cent(7)
certify mass storage media	mediainit(1)
cfgetispeed() - get tty input baud rate	cfspeed(3C)
cfgetospeed() - get tty output baud rate	cfspeed(3C)
cf1 - configure a SCSI disk array LUN	cf1(1M)
cfsetispeed() - set tty input baud rate	cfspeed(3C)
cfsetospeed() - set tty output baud rate	cfspeed(3C)
chac1 - add, modify, delete, copy, or summarize file access control lists (ACLs)	chac1(1)
change (delta) to an SCCS file, make a	delta(1)
change access rights on an NIS+ object	nischmod(1)
change active controllers on HP-IB	hpib_pass_ctl(3I)
change attributes of a DOS file	doschmod(1)
change characteristics of physical volume in LVM volume group	pvchange(1M)
change current login to a new group	newgrp(1)
change data format of and copy a (tape) file	dd(1)
change data segment space allocation	brk(2)
change default login shell	chsh(1)
change delta commentary of an SCCS delta	cdc(1)
change file mode access permissions	chmod(1)
change file mode access permissions	chmod(2)
change file owner or group	chown(1)
change global search path for dynamically loadable kernel modules	modpath(2)
change login name	su(1)
change login password in Network Information System (NIS)	yppasswd(1)
change login password	passwd(1)
change LVM logical volume characteristics	lvchange(1M)

Index

All Volumes

Description	Entry Name(Section)
change machine information	setuname(1M)
change NIS+ password information	nispasswd(1)
change or add value to environment	putenv(3C)
change or display event or system call audit status	audevent(1M)
change or examine blocked signals	sigprocmask(2)
change or examine signal action	sigaction(2)
change or examine signal action	sigwait(2)
change or query stream configuration	strchg(1M)
change or reformat a text file	newform(1)
change owner and group of a file	chown(2)
change owner and/or group in access control list (ACL)	chownacl(3C)
change owner of an NIS+ object	nischown(1)
change priority of a process	nice(2)
change priority of running processes	renice(1)
change program's internal attributes	chatr(1)
change RCS file attributes	rcs(1)
change real-time priority	rtprio(2)
change renditions of characters in a window	chgat(3X)
change root directory	chroot(2)
change root directory for a command	chroot(1M)
change selected characters	tr(1)
change state, wait for child process to	wait3(2)
change state, wait for child process to	waitid(2)
change system configuration file	ch_rc(1M)
change the default stacksize	pthread_default_stacksize_np(3T)
change the group owner of an NIS+ object	nischgrp(1)
change the name of a file	mv(1)
change the name of a file	rename(2)
change the signal mask of the calling thread	pthread_sigmask(3T)
change the time to live value of an NIS+ object	nischttl(1)
change user information in password file; used by finger	chfn(1)
change user's secure RPC key	chkey(1)
change working directory	cd(1)
change working directory	chdir(2)
change, access, and/or modification times of a file, update	touch(1)
changer device driver, SCSI media	autochanger(7)
changes NIS information	ypupdate(3C)
changing NIS information, server for	ypupdated(1M)
channel from buffers, perform I/O with an HP-IB	hplib_io(3I)
channel, create an interprocess	pipe(2)
character and rendition to a window, add a complex	add_wch(3X)
character and rendition, complex, input from a window	in_wch(3X)
character and rendition, complex, insert into a window	ins_wch(3X)
character and rendition, input a single-byte from a window	inch(3X)
character and rendition, single-byte, insert into a window	insch(3X)
character back into input stream, push	ungetc(3S)
character code set, convert to another	iconv(1)
character code set, convert to another	iconv(3C)
character device special file, control	ioctl(2)
character	glossary(9)
character or data word from a stream file, get	getc(3S)
character or word, put on a stream	putc(3S)
character rendition, write and immediately refresh the pad	pechochar(3X)
character sequences for display/keyboard, convert file data order	forder(1)
character set	glossary(9)
character special file	glossary(9)
character string and rendition, wide, get from a cchar_t	getcchar(3X)
character string operations	string(3C)
character string or stream file, read from with formatted input conversion	scanf(3S)
character string, multi-byte, input from a window	innstr(3X)
character, generate printable representation of	unctrl(3X)

Description	Entry Name(Section)
character, get a multi-byte character length limited string from the terminal	getnstr(3X)
character, get a multi-byte character string from the terminal	getstr(3X)
character, get a wide character from a terminal	get_wch(3X)
character, insert a wide-character string into a window	ins_nwstr(3X)
character, multi-byte, insert into a window	insnstr(3X)
character, push onto the input queue	ungetch(3X)
character, single-byte, get from the terminal	getch(3X)
character-string login name of the user, get	cuserid(3S)
Character-Terminal User Environment (CUE), HP	cue(1)
characteristics for cue , set terminal	cuegetty(1M)
characteristics of a disk device, describe	diskinfo(1M)
characteristics of nodes on FDDI ring, list	fddinet(1M)
characteristics of physical volume in LVM volume group, change	pvchange(1M)
characteristics, change LVM logical volume	lvchange(1M)
characters and renditions, an array of single-byte, input from a window	inchnstr(3X)
characters and renditions, complex, draw lines from	hline_set(3X)
characters and renditions, draw lines from single-byte	hline(3X)
characters and strings conversions, multibyte	multibyte(3C)
characters in a file, count	wc(1)
characters in a file, unprintable and non-ASCII, make visible or invisible	vis(1)
characters, alter, delete, modify, substitute, or translate	tr(1)
characters, ASA carriage control, interpret	asa(1)
characters, classify according to type	cctype(3C)
characters, classify according to type	wctype(3C)
characters, how to type control	ascii(5)
characters, renditions of, change in a window	chgat(3X)
characters, tools to process 16-bit	nl_tools_16(3X)
characters, translate to upper-case, lower-case, or 7-bit ASCII	conv(3C)
characters, wide, input a string of, from a window	innwstr(3X)
CHARADV() - get character and advance pointer to next character	nl_tools_16(3X)
CHARAT() - get value of 8- or 16-bit character	nl_tools_16(3X)
chargefee - charge fee to user based on system usage	acctsh(1M)
charmap - symbolic translation file for localedef scripts	charmap(4)
chatr - change program's internal attributes	chatr(1)
chdir - change current working directory	csh(1)
chdir() - change working directory	chdir(2)
check and repair VxFS file system	fsck_vxfs(1M)
check file system consistency and interactively repair	fsck(1M)
check file system consistency and interactively repair (generic)	fsck(1M)
check if system has been converted to a trusted system	iscomsec(2)
check in RCS revisions	ci(1)
check internal consistency of Authentication database	authck(1M)
check internal revision numbers of HP-UX files	revck(1M)
check nroff/troff files	checknr(1)
check or print documents formatted with the mm macros	mm(1)
check or repair a physical volume in LVM volume group	pvck(1M)
check out RCS revisions	co(1)
check password or group file	pwck(1M)
check Product Description File against file system	pdfck(1M)
check the network, scatter data to	spray(3N)
check the uucp directories and permissions file	uuccheck(1M)
checker, generic file system quota consistency	quotacheck(1M)
checker, hfs file system quota consistency	quotacheck_nfs(1M)
checker, VxFS file system quota consistency	quotacheck_vxfs(1M)
checking VxFS file system with label	volcopy_vxfs(1M)
checking, copy file systems with label	volcopy(1M)
checking, copy file systems with label	volcopy_hfs(1M)
checknr - check nroff/troff files	checknr(1)
checksum and block count of a file, print	sum(1)
checksum and block count of a file, print	sum(1)
cheduling policy and associated parameters, get and set	pthread_getschedparam(3T)

Index

All Volumes

Description	Entry Name(Section)
chfn - change user information in password file; used by finger	chfn(1)
chgat () - change renditions of characters in a window	chgat(3X)
chgrp - change group of file	chown(1)
child process and process times, get	times(2)
child process	fork(2)
child process	glossary(9)
child process to change state, wait for	wait3(2)
child process to stop or terminate, wait for	wait(2)
child process, wait to change state	waitid(2)
children, synchronise a window with	syncok(3X)
chkey - change user's secure RPC key	chkey(1)
chmod - change file mode access permissions	chmod(1)
chmod () - change file mode access permissions	chmod(2)
chown - change file owner	chown(1)
chown () - change owner and group of a file	chown(2)
chownacl () - change owner and/or group in access control list (ACL)	chownacl(3C)
chroot - change root directory for a command	chroot(1M)
chroot () - change root directory	chroot(2)
chsh - change default login shell	chsh(1)
ch_rc - change system configuration file	ch_rc(1M)
ci - check in RCS revisions	ci(1)
circuit, X.25 switched virtual, clear	clrsvc(1M)
ckpacct - check size of process accounting file	acctsh(1M)
cksum - print file checksum and sizes	sum(1)
class-dependent data translation of ELF files	elf_xlate(3E)
class-dependent object file header for ELF files, retrieve	elf_getehdr(3E)
class-dependent program header table for ELF files, retrieve	elf_getphdr(3E)
class-dependent section header for ELF files, retrieve	elf_getshdr(3E)
classification macro, floating-point	fpclassify(3M)
classify characters according to type	ctype(3C)
classify characters according to type	wctype(3C)
clean file system at last system shutdown, test for	fsclean(1M)
clean-up, uucp spool directory	uucleanup(1M)
clear a window	clear(3X)
clear - clear terminal screen	clear(1)
clear from cursor to end of line	clrtoeol(3X)
clear from cursor to end of window	clrtobot(3X)
clear inode	clri(1M)
clear or set auditing on calling process	setaudproc(2)
clear the process environment	clearenv(3C)
clear window attributes	standend(3X)
clear X.25 switched virtual circuit	clrsvc(1M)
clear () - clear a window	clear(3X)
clearenv () - clear the process environment	clearenv(3C)
clearerr () - clear I/O error on stream	ferror(3S)
clearok () - terminal output control functions	clearok(3X)
client and server, NIS+, initialization utility	nisinit(1M)
client configuration information file, diskless	info(4)
CLIENT handles, library routines for dealing with creation and manipulation of	rpc_clnt_create(3N)
client interface, Network Information Service	ypclnt(3C)
client side, library routines for client side calls, rpc	rpc_clnt_calls(3N)
client, library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
clients, directories to export to NFS	exports(4)
clients, export and unexport directories to NFS	exportfs(1M)
clients, PFS, directories to export to	pfs_exports(5)
clients, PFS, export and unexport directories to	pfs_exports(1M)
clntraw_create () - obsolete library routines for RPC	rpc_soc(3N)
clnttcp_create () - obsolete library routines for RPC	rpc_soc(3N)
clntudp_bufcreate () - obsolete library routines for RPC	rpc_soc(3N)
clntupd_create () - obsolete library routines for RPC	rpc_soc(3N)
clnt_broadcast () - obsolete library routines for RPC	rpc_soc(3N)

Description	Entry Name(Section)
clnt_call() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_control() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_create_vers() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_destroy() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_dg_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_freeres() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_geterr() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_pcreateerror() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_perrno() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_perror() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_raw_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_spccreateerror() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_sperrno() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_sperror() - library routines for client side calls	rpc_clnt_calls(3N)
clnt_tli_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_tp_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clnt_vc_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
clock daemon	cron(1M)
clock date and time, get system	gettimeofday(2)
clock operations	clocks(2)
clock resolution, get	clocks(2)
clock tick	glossary(9)
clock time value, get	clocks(2)
clock time value, set	clocks(2)
clock() - report CPU time used	clock(3C)
clock, correct the time to synchronize the system	adjtime(2)
clock, get current value of system-wide	getclock(3C)
clock, set value of system-wide	setclock(3C)
clock, system, display current date and time or set to new value	date(1)
clocks - clock operations	clocks(2)
clock_getres() - get clock resolution	clocks(2)
clock_gettime() - get clock time value	clocks(2)
clock_settime() - set clock time value	clocks(2)
clone driver: STREAMS driver	clone(7)
clone - open a major and minor device pair on a STREAMS driver	clone(7)
close a crash dump descriptor	cr_close(3)
close a message queue descriptor	mq_close(2)
close a named semaphore	sem_close(2)
close a shared object	dlclose(3C)
close a stream	fclose(3S)
close legal user shells file	getusershell(3C)
close or open NLS message catalog for reading	catopen(3C)
close or open pipe I/O to or from a process	popen(3S)
close transport endpoint (X/OPEN TLI-XTI)	t_close(3)
close() - close a file descriptor	close(2)
close, access, or open a directory and associated directory stream	directory(3C)
closedir() - close a currently open directory	directory(3C)
closelog() - close system log file	syslog(3C)
clri - clear inode	clri(1M)
clrsvc - clear X.25 switched virtual circuit	clrsvc(1M)
clrtobot() - clear from cursor to end of window	clrtobot(3X)
clrtoeol() - clear from cursor to end of line	clrtoeol(3X)
cluster, write a message simultaneously to all users	wall(1M)
clusters, DOS disk, report number of free	dosdf(1)
cmp - compare two files	cmp(1)
co - check out RCS revisions	co(1)
code files, object, in a library, find optimum sequence for	lorder(1)
code set conversion, character	iconv(1)
code set conversion, character	iconv(3C)
code widths, set and get EUC for ldterm	eucset(1)

Index

All Volumes

Description	Entry Name(Section)
code, processor-dependent (firmware)	pdc(1M)
coded character set	glossary(9)
col - filter reverse line-feeds and backspaces from text	col(1)
collating element	glossary(9)
collation	glossary(9)
collation sequence	glossary(9)
collect system diagnostic messages to form error log	dmesg(1M)
color manipulation functions	can_change_color(3X)
color_content() - color manipulation functions	can_change_color(3X)
color_set() - window attribute control functions	attr_get(3X)
COLS() - number of columns on terminal screen	COLS(3X)
columns, number of, on terminal screen	COLS(3X)
comb - combine SCCS deltas	comb(1)
combine corresponding lines of several files or subsequent lines of one file	paste(1)
combine SCCS deltas	comb(1)
combine two LVM logical volumes into one logical volume	lvmerge(1M)
comm - select/reject lines common to two sorted files	comm(1)
command bytes over HP-IB, send	hpib_send_cmnd(3I)
command - execute a simple command	command(1)
command execution, set (modify or redefine) environment for	env(1)
command execution, UNIX system to UNIX system	uux(1)
command	glossary(9)
command history and input editor for interactive programs	ied(1)
command interpreter (shell) with C-like syntax	csh(1)
command interpreter	glossary(9)
command line tool for DHCP elements of bootpd	dhcptools(1M)
command name, find program files that execute under given	which(1)
command on a remote host, execute a	rcmd(3N)
command on a remote host, execute	on(1)
command options, parse	getopt(1)
command options, parse	getopts(1)
Command Set 1980	glossary(9)
Command Set 80 Cartridge Tape I/O Utility	tcio(1)
command summary from per-process accounting records	acctcms(1M)
command usage summary and accounting data, accumulate	acctsh(1M)
command, change root directory	chroot(1M)
command, execute a simple	command(1)
command, measure time used to execute a	time(1)
command, remote, return a stream to	rcmd(3N)
command, report execution time of, process accounting data and system activity	timex(1)
command, return stream to a remote	rexec(3N)
command, run at nondefault priority	nice(1)
command, run immune to hangsups	nohup(1)
command, shell, issue a	system(3S)
commands to Terminal Session Manager, send	tsm.command(1)
commands, execute at a later time	at(1)
commands, file system administration configuration and binary files	fs_wrapper(5)
commands, generic I/O device control	ioctl(5)
commands, install new	install(1M)
commands, output to the terminal	putp(3X)
commands, RCS, description of	rcsintro(5)
commands, show last executed in reverse order	lastcomm(1)
commands: STREAMS ioctl commands	streamio(7)
commentary of an SCCS delta, change delta	cdc(1)
common archive file format	ar(4)
common logarithm functions	log10(3M)
common to two sorted files, reject/select lines	comm(1)
communicate interactively with another user	write(1)
communication domain protocol, local	UNIX(7P)
communication facilities, interprocess, report status	ipcs(1)
communication identifier, create interprocess	ftok(3C)

Description	Entry Name(Section)
communication software for serial and network connections	kermit(1)
communication, create an endpoint for	socket(2)
communications, Interprocess	socket(7)
communications, Interprocess	xopen_networking(7)
compact - compact files using Huffman code (see <i>pack(1)</i>)	compact(1)
compact files using Huffman code (see <i>pack(1)</i>)	compact(1)
compact list of users currently on the system	users(1)
compaction, copy HFS file system with	dcopy(1M)
compare contents of memory with byte	memory(3C)
compare contents of two directories	dircmp(1)
compare or print out terminfo descriptions	infocmp(1M)
compare Product Description File and file system	pdfck(1M)
compare RCS revisions	rcsdiff(1)
compare sorted files; reject/select common lines	comm(1)
compare three files and find differences	diff3(1)
compare two files and find differences	diff(1)
compare two files and mark differences	diffmk(1)
compare two files and show differences side-by-side	sdiff(1)
compare two files	cmp(1)
compare two strings	string(3C)
compare two thread identifiers	pthread_equal(3T)
compare two versions of an SCCS file	sccsdiff(1)
compare two wide strings	wcstring(3C)
comparison macro, floating-point (<)	isless(3M)
comparison macro, floating-point (<=)	islessequal(3M)
comparison macro, floating-point (<=)	islessgreater(3M)
comparison macro, floating-point (>)	isgreater(3M)
comparison macro, floating-point (>=)	isgreaterequal(3M)
comparison macro, floating-point (unordered)	isunordered(3M)
comparison routines for regular expressions	regexp(3X)
compile a regular expression	regcmp(3X)
compile and execute regular expressions	re_comp(3X)
compile and match routines for regular expressions	regexp(3X)
compile() - regular expression compile routine	regexp(3X)
compiled terminfo file format	term(4)
compiler, bundled C	cc_bundled(1)
compiler/interpreter for modest-sized programs	bs(1)
compilers, rpcgen; generate RPC protocols, C header files	rpcgen(1)
compilers: terminfo data base compiler	tic(1M)
compiling routines, regular expression	regcomp(3C)
complementary error function	erf(3M)
complete, wait for background processes to	wait(1)
complex character and rendition, add to a window	add_wch(3X)
complex character and rendition, input from a window	in_wch(3X)
complex character and rendition, insert into a window	ins_wch(3X)
complex character, set or get background character and rendition using	bkgrnd(3X)
complex character, write and immediately refresh the window	echo_wchar(3X)
complex characters and renditions, add an array of, to a window	add_wchnstr(3X)
complex characters and renditions, draw borders	border_set(3X)
complex characters and renditions, draw borders from	box_set(3X)
complex characters and renditions, draw lines from	hline_set(3X)
complex characters and renditions, input an array of, from a window	in_wchnstr(3X)
composite graphic symbol	glossary(9)
compress , uncompress , zcat - compress or expand data	compress(1)
compress or expand data	compress(1)
compressdir , uncompressdir - compress or expand files in a directory	compress(1)
compute hash value for ELF files	elf_hash(3E)
compute shortest path and route between hosts	pathalias(1)
computer system information, display	uname(1)
computer system information, get	uname(2)
computer system, set node name	uname(1)

Index All Volumes

Description	Entry Name(Section)
computer system, set node name	uname(2)
concatenate two strings	string(3C)
concatenate two wide strings	wcstring(3C)
concatenate, copy, and print files	cat(1)
concurrency level of unbound threads, get and set	pthread_getconcurrency(3T)
condition becomes true, wait until the requested status	hplib_status_wait(3I)
condition variable attributes object, initialize or destroy	pthread_condattr_init(3T)
condition variable, unblock one or all threads waiting on a conditional variable	pthread_cond_signal(3T)
condition variable, wait or timed wait on	pthread_cond_wait(3T)
condition, evaluate for true/false	test(1)
conditional expression: return the state of the NIS+ namespace	nistest(1)
conditions, define for I/O device interrupt	io_on_interrupt(3I)
conduct a serial poll on HP-IB	hplib_spoll(3I)
config - configure and build an HP-UX system	config(1M)
configurable path name variables, get	pathconf(2)
configurable system variables, get	sysconf(2)
configuration and loadable flags for a module, set and query	kmsystem(1M)
configuration and status, display LAN device	lanscan(1M)
configuration backup file, create or update LVM volume group	vgcfgbackup(1M)
configuration database, network	netconfig(4)
configuration file for HPDPS gateway printers	pdgwcfg.conf(4)
configuration file for inetd	inetd.conf(4)
configuration file for NIS updating	updaters(1M)
configuration file for pluggable authentication module	pam.conf(4)
configuration file for secure internet services	inetsvcs.conf(4)
configuration file for the name-service switch	nsswitch.conf(4)
configuration file for the SNMP agent	snmpd.conf(4)
configuration file, change system	ch_rc(1M)
configuration file, network tracing and logging	nettlgen.conf(4)
configuration file, resolver	resolver(4)
configuration file, used by DDFA software	pcf(4)
configuration files used by file system administration commands	fs_wrapper(5)
configuration information file, diskless client	info(4)
configuration information tool, multicast routing	mrinfo(1M)
configuration table for disk arrays	arraytab(4)
configuration table, boot device	bootconf(4)
configuration tool, EISA	eisa_config(1M)
configuration values, get POSIX	getconf(1)
configuration values, get string-valued	confstr(3C)
configuration, gated	gated.conf(4)
configuration, master kernel configuration information	master(4)
configuration, restore volume group	vgcfgrestore(1M)
configuration, user, files for pluggable authentication modules	pam_user.conf(4)
configure and build an HP-UX system	config(1M)
configure network interface parameters	ifconfig(1M)
configure network tracing and logging subsystem database	nettlconf(1M)
configure system crash dumps	crashconf(1M)
configure system crash dumps	crashconf(2)
configure system language on multi-language systems	geocustoms(1M)
configure the LP spooling system	lpadmin(1M)
configure, software products	swinstall(1M)
configuring a disk array	newarray(1M)
confirmation from connect request (X/OPEN TLI-XTI)	t_rcvconnect(3)
confstr() - get string-valued configuration values	confstr(3C)
connect accounting records, manipulate	fwtmp(1M)
connect all system FDDI network interfaces, initialize and	fddisetup(1M)
connect request (X/OPEN TLI-XTI)	t_listen(3)
connect request issued by a transport user (X/OPEN TLI-XTI)	t_accept(3)
connect to the FDDI network	fddiinit(1M)
connect() - initiate connection on socket	connect(2)
connected peer, get address of	getpeername(2)

Description	Entry Name(Section)
connected sockets, create a pair	socketpair(2)
connection daemon debug utility used by DDFA software, outbound	ocdebug(1M)
connection daemon used by DDFA software, outbound	ocd(1M)
connection mapper, multicast router	map-mbone(1M)
connection on a socket, accept	accept(2)
connection on a socket, initiate	connect(2)
connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
connection, establish an out-bound terminal line	dial(3C)
connection, receive data (X/OPEN TLI-XTI)	t_rcv(3)
connection, send data (X/OPEN TLI-XTI)	t_snd(3)
connections on a socket, listen for	listen(2)
connectivity, verify LAN with link-level loopback	linkloop(1M)
consistency checker, generic file system quota	quotacheck(1M)
consistency checker, hfs file system quota	quotacheck_nfs(1M)
consistency checker, VxFS file system quota	quotacheck_vxfs(1M)
consistency of Authentication database, check internal	authchk(1M)
consistency, maintain between kernel and /etc/ioconfig	ioinit(1M)
console and standard error, displays formatted message on	fntmsg(3C)
console - system console interface special file	console(7)
console, search for during boot process	pdc(1M)
constants and values for programming, machine-dependent	values(5)
constants, implementation-specific	limits(5)
constants, language information	langinfo(5)
constants, math functions and	math(5)
constants, standard structures and symbolic	unistd(5)
construct a file system (generic)	mkfs(1M)
construct a new file system (generic)	newfs(1M)
construct a new HFS file system	newfs_hfs(1M)
construct an HFS file system	mkfs_hfs(1M)
construct argument list(s) and execute command	xargs(1)
construct new VxFS file system	newfs_vxfs(1M)
construct VxFS file system	mkfs_vxfs(1M)
constructs, nroff/troff, tbl, and neqn, remove	deroff(1)
consumption limit, get or set system resource	getrlimit(2)
contents of an NIS+ directory: list	nisl(1)
contents of directory, list	ls(1)
contents of DOS directories, list	dosls(1)
contents of the NIS+ transaction log, display	nislog(1M)
contents of two directories, compare	dircmp(1)
context, signal stack, set and/or get	sigstack(2)
context-dependent file	chmod(1)
context-dependent file	chmod(2)
context-sensitive softkey shell	keysh(1)
continue - resume execution of nearest while or foreach	csh(1)
continue - resume next iteration of enclosing for/next loop	ksh(1)
continue - resume next iteration of enclosing for/next loop	sh-bourne(1)
continue - resume next iteration of enclosing for/next loop	sh-posix(1)
continue, resume, or suspend execution of a thread	pthread_resume_np(3T)
control a file descriptor for ELF files	elf_ctl(3E)
control a SCSI device	scsictl(1M)
control address resolution	arp(1M)
control blocking on input	notimeout(3X)
control character device special file	ioctl(2)
control character	glossary(9)
control characters, ASA carriage, interpret	asa(1)
control characters, how to type	ascii(5)
control checking for typeahead	typeahead(3X)
control commands, generic I/O device	ioctl(5)
control device driver, SCSI device	scsi_ctl(7)
control device driver, SCSI device	scsi_pt(7)
control EOI mode for HP-IB file	hpiib_eoi_ctl(3I)

Index

All Volumes

Description

Entry Name(Section)

control function, for window refresh	touchwin(3X)
control functions for window attribute	attr_get(3X)
control functions, input mode	cbreak(3X)
control functions, restricted window attribute	attroff(3X)
control functions, terminal output	clearok(3X)
control functions, tty line	tccontrol(3C)
control functions, VxFS file system	vxfsio(7)
control functions, window refresh	is_linetouched(3X)
control input character delay mode	halfdelay(3X)
control lines on GPIO card, set	gpio_set_ctl(3I)
control network tracing and logging	nettl(1M)
control operations, message	msgctl(2)
control operations, semaphore	semctl(2)
control operations, shared memory	shmctl(2)
control panel, audio	see audio(5)
control response to parallel poll on HP-IB	hpib_card_ppoll_resp(3I)
control routines for open-files	fcntl(2)
control system log	syslog(3C)
control system resource consumption limit	getrlimit(2)
control terminal device (Bell Version 6 compatibility)	stty(2)
control the HP-IB interface Remote Enable line	hpib_ren_ctl(3I)
control tty device	tcattribute(3C)
control, asynchronous serial modem lines	modem(7)
control, file system	fsctl(2)
control, multiprocessor	mpctl(2)
control, uucp status inquiry and job	uustat(1)
control, version	vc(1)
controllers on HP-IB, change active	hpib_pass_ctl(3I)
controlling process	glossary(9)
controlling terminal	glossary(9)
controlling terminal, generate file name of	ctermid(3S)
conv() , - translate characters	conv(3C)
conventions, file-name suffix	suffix(5)
conventions, numeric formatting, of current locale, query	localeconv(3C)
conversion function, string-to-NaN	nan(3M)
conversion routines, network station address string	net_aton(3C)
conversion, date and time	strptime(3C)
conversion, formatted input, to a varargs argument	vscanf(3S)
conversions, multibyte characters and strings	multibyte(3C)
convert 9-digit hash codes to compressed spelling reference list	spell(1)
convert a file system to allow long file names	convertfs(1M)
convert access control list (ACL) structure to string form	acltostr(3C)
convert between 3-byte integers and long integers	l3tol(3C)
convert between long integer and base-64 ASCII string	a64l(3C)
convert binary file to ASCII for transmission by mailer	uuenocode(1)
convert character code set to another	iconv(1)
convert character code set to another	iconv(3C)
convert - convert audio file	convert(1)
convert date and time to string	ctime(3C)
convert date and time to string	strftime(3C)
convert date and time to wide-character string	wcsftime(3C)
convert DOS file to HP-UX ASCII file format	dos2ux(1)
convert file keyboard/display data order	forder(1)
convert file to stream	fopen(3S)
convert floating-point number to string or string array element	ecvt(3C)
convert formatted input from a window	mvscanw(3X)
convert formatted input from a window	vwscanw(3X)
convert formatted input from a window	vw_scanw(3X)
convert HP-UX ASCII file to DOS file format	dos2ux(1)
convert login/logoff records to per-session accounting records	acctcon(1M)
convert long double floating-point number to string	ldcv(3C)

Description	Entry Name(Section)
convert long integer to string	ltostr(3C)
convert per-session records to total accounting records	acctcon(1M)
convert spelling reference list words to 9-digit hash codes for spell	spell(1)
convert string data order	strord(3C)
convert string to access control list (ACL) structure	strtoacl(3C)
convert string to double-precision number	strtod(3C)
convert string to long double-precision number	strtold(3C)
convert tabs to spaces, and vice versa	expand(1)
convert text words to 9-digit hash codes for spell	spell(1)
convert underscores to underlining on terminal	ul(1)
convert units of measure	units(1)
convert user format date and time	getdate(3C)
convert values between host and network byte order	byteorder(3N)
convert wide character string to double-precision number	wcstod(3C)
convert wide character string to long integer	wcstol(3C)
convert, reblock, translate, and copy a (tape) file	dd(1)
converted trusted system, check if	iscomsec(2)
convertfs - convert a file system to allow long file names	convertfs(1M)
converts old sendmail.cf files to new format	convert_awk(1M)
convert_awk - converts old sendmail.cf files to new format	convert_awk(1M)
coordinate ELF library and application versions	elf_version(3E)
coordinate, window, transformation, define	mvderwin(3X)
Coordinated Universal Time (UTC)	date(1)
Coordinated Universal Time (UTC)	glossary(9)
coordinates, get additional cursor and window coordinates	getbegyx(3X)
coordinates, get cursor and window coordinates	getyx(3X)
copy a file into memory	copylist(3C)
copy a region of window	copywin(3X)
copy access control list (ACL) to another file	cpacl(3C)
copy displayed CRT terminal output simultaneously in a file	script(1)
copy file archives in and out	cpio(1)
copy file systems with label checking	volcopy(1M)
copy file systems with label checking	volcopy_hfs(1M)
copy file to a new or existing file	cp(1)
copy file, public UNIX system to UNIX system	uuto(1)
copy files between systems	ftp(1)
copy files to or from remote system	rcp(1)
copy files to or from remote system	rcp(1)
copy HFS file system with compaction	dcopy(1M)
copy in, copy out - transfer uucp -system files in or out	uucico(1M)
copy memory to another area	memory(3C)
copy multiple files to a directory	cp(1)
copy of standard output, send to specified file	tee(1)
copy overlapped windows	overlay(3X)
copy to or from DOS files	doscp(1)
copy to or from LIF files	lifcp(1)
copy unwritten system buffers to disk	sync(1M)
copy unwritten system buffers to disk periodically	syncer(1M)
copy VxFS file system with label checking	volcopy_vxfs(1M)
copy, add, modify, delete, or summarize file access control lists (ACLs)	chacl(1)
copy, concatenate, and print files	cat(1)
copy, software products	swinstall(1M)
copydvagent() - copy device assignment structure	getdvagent(3)
copylist() - copy a file into memory	copylist(3C)
copysign functions	copysign(3M)
copysign() , copysignf() - copysign functions	copysign(3M)
copysignf() , copysign() - copysign functions	copysign(3M)
copywin() - copy a region of window	copywin(3X)
core - core image file format	core(4)
core image file format	core(4)
correct the time to synchronize the system clock	adjtime(2)

Index

All Volumes

Description	Entry Name(Section)
<code>cos()</code> - cosine function	<code>cos</code> (3M)
<code>cosd()</code> - cosine function (degrees)	<code>cosd</code> (3M)
<code>cosdf()</code> - cosine function (float, degrees)	<code>cosd</code> (3M)
<code>cosf()</code> - cosine function (float)	<code>cos</code> (3M)
<code>cosh()</code> , <code>coshf()</code> - hyperbolic cosine functions	<code>cosh</code> (3M)
<code>coshf()</code> , <code>cosh()</code> - hyperbolic cosine functions	<code>cosh</code> (3M)
cosine function, inverse hyperbolic	<code>acosh</code> (3M)
cosine functions	<code>cos</code> (3M)
cosine functions, hyperbolic	<code>cosh</code> (3M)
cosine trigonometric function (degrees)	<code>cosd</code> (3M)
count adjacent repeated lines in a file	<code>uniq</code> (1)
count words, lines, and bytes or characters in a file	<code>wc</code> (1)
<code>cp</code> - copy file, files, or directory subtree	<code>cp</code> (1)
<code>cpacl()</code> - copy access control list (ACL) to another file	<code>cpacl</code> (3C)
<code>cpio</code> archive format	<code>cpio</code> (4)
<code>cpio</code> - copy file archives in and out	<code>cpio</code> (1)
<code>cpio</code> - format of <code>cpio</code> archive	<code>cpio</code> (4)
<code>cpp</code> - the C language preprocessor	<code>cpp</code> (1)
<code>cpset</code> - install object files in binary directories	<code>cpset</code> (1M)
CPU time used, report	<code>clock</code> (3C)
<code>cpu</code> , set name of host	<code>sethostname</code> (2)
crash dump access library	<code>libcrash</code> (5)
crash dump data, manipulate	<code>crashutil</code> (1M)
crash dump descriptor, close	<code>cr_close</code> (3)
crash dump information, retrieve	<code>cr_info</code> (3)
crash dump of the operating system, save	<code>savecrash</code> (1M)
crash dump, open for reading	<code>cr_open</code> (3)
crash dump, read from	<code>cr_read</code> (3)
crash dump, verify integrity of	<code>cr_verify</code> (3)
crash dumps, configure system	<code>crashconf</code> (1M)
crash dumps, configure system	<code>crashconf</code> (2)
crash	<code>glossary</code> (9)
crash, patch up damaged file system (generic)	<code>fsdb</code> (1M)
crash, patch up damaged HFS file system	<code>fsdb_hfs</code> (1M)
<code>crashconf</code> - configure system crash dumps	<code>crashconf</code> (1M)
<code>crashconf()</code> - configure system crash dumps	<code>crashconf</code> (2)
<code>crashutil</code> - manipulate crash dump data	<code>crashutil</code> (1M)
<code>creat()</code> - create a new file or rewrite an existing one	<code>creat</code> (2)
<code>creat64()</code> - file system API to support large files	<code>creat64</code> (2)
create a directory file	<code>mkdir</code> (2)
create a directory	<code>mkdir</code> (1)
create a directory, special, or ordinary file	<code>mknod</code> (2)
create a DOS directory	<code>dosmkdir</code> (1)
create a makefile	<code>mkmf</code> (1)
create a name for a temporary file	<code>tmpnam</code> (3S)
create a new file	<code>creat</code> (2)
create a new file or rewrite an existing one	<code>creat</code> (2)
create a new key in the <code>publickey</code> database file	<code>newkey</code> (1M)
create a new process	<code>fork</code> (2)
create a new thread of execution	<code>pthread_create</code> (3T)
create a pair of connected sockets	<code>socketpair</code> (2)
create a socket	<code>socket</code> (2)
create a special (device) file	<code>mksf</code> (1M)
create a tags file	<code>ctags</code> (1)
create a temporary file	<code>tmpfile</code> (3S)
create a unique (usually temporary) file name	<code>mktemp</code> (3C)
create an endpoint for communication	<code>socket</code> (2)
create an interprocess channel	<code>pipe</code> (2)
create and administer SCCS files	<code>admin</code> (1)
create and monitor jobs	<code>swjob</code> (1M)
create backup LVM volume group configuration file	<code>vgcfgbackup</code> (1M)

Description	Entry Name(Section)
create empty administration file owned by adm with mode 664	acctsh (1M)
create file names	glob (3C)
create interprocess communication identifier	ftok (3C)
create logical volume in LVM volume group	lvcreate (1M)
create LVM volume group	vgcreate (1M)
create message catalog file for modification	findmsg (1)
create message files for use by gettext()	mkmsgs (1)
create NIS+ credentials	nisaddcred (1M)
create NIS+ directories	nismkdir (1)
create NIS+ tables from corresponding /etc files or NIS maps	nisaddent (1M)
create or destroy a thread-specific data key	pthread_key_create (3T)
create or rebuild Network Information Service database	ypmake (1M)
create periodic accounting summary files	acctsh (1M)
create physical volume for use in LVM volume group	pvcreate (1M)
create Product Description File from an input	mkpdf (1M)
create sendmail alias file	sendmail (1M)
create session and set process group ID	setsid (2)
create special and FIFO files	mknod (1M)
create the cat files for the manual	catman (1M)
create user crontab file	crontab (1)
create zero-length file	cat (1)
create zero-length file	cp (1)
create zero-length file	null (7)
create zero-length file	touch (1)
create, distribute, install, monitor, and manage software	sd (5)
create, remove directories in a path	mkdirp (3G)
create, windows, functions	newwin (3X)
create/open a message queue	mp_open (2)
create/open a shared memory object	shm_open (2)
creates database maps for sendmail	makemap (1M)
creates print objects	pdcreate (1)
create_sysfile - create a kernel system file	create_sysfile (1M)
creating PAM sessions	pam_open_session (3)
creation function, relative window	derwin (3X)
credentials, NIS+, initialize for NIS+ principals	nisclient (1M)
crontab , batch , and at queue description file	queuedefs (4)
crontab file operations, user	crontab (1)
crontab - user crontab file operations	crontab (1)
CRT or line-printer output, format text file for	nroff (1)
CRT terminal, record displayed output simultaneously in a file	script (1)
CRT terminals, peruse file on	more (1)
crt0.o , gcr0.o , mcrt0.o , frt0.o , mfrt0.o - execution startup routines	crt0 (3)
crt0.o , mcrt0.o - C and Pascal execution startup routines	crt0 (3)
crypt - encode/decode files	crypt (1)
crypt() , setkey() , setkey_r() , encrypt_r() , encrypt() - generate hashing encryption	crypt (3C)
cr_close() - close a crash dump descriptor	cr_close (3)
cr_info() - retrieve crash dump information	cr_info (3)
cr_isaddr() - validate whether physical page number was dumped	cr_isaddr (3)
cr_open() - open crash dump for reading	cr_open (3)
cr_perror() - print a libcrash error or warning message	cr_perror (3)
cr_read() - read from crash dump	cr_read (3)
cr_uncompress() - uncompress a file in a crash dump	cr_uncompress (3)
cr_verify() - verify integrity of crash dump	cr_verify (3)
CS-80	glossary (9)
CS/80 Cartridge Tape I/O Utility	tcio (1)
CS/80	glossary (9)
csh - a shell (command interpreter) with C-like syntax	csh (1)
csplit - context split	csplit (1)
ct - cartridge tape device access drivers	ct (7)
ct - spawn getty to remote terminal (call terminal)	ct (1)
ctags - create a tags file	ctags (1)

Description	Entry Name(Section)
ctermid() - generate file name for terminal	ctermid(3S)
ctime and date, time zone adjustment table for	tztab(4)
ctime(), ctime_r() - convert clock() date and time to string	ctime(3C)
cu - call another (UNIX) system; terminal emulator	cu(1)
cube root functions	cbrt(3M)
cue, set terminal characteristics for	cuegetty(1M)
cue - HP Character-Terminal User Environment (CUE)	cue(1)
cuegetty - set terminal characteristics for cue	cuegetty(1M)
current absolute system time, add a specific time interval to the	get_expiration_time(3T)
current directory	glossary(9)
current erase and line kill characters	erasewchar(3X)
current events and system calls to be audited	setevent(2)
current host system, set or display name of	hostname(1)
current host, get an identifier for	gethostid(2)
current host, get name of	gethostname(2)
current HP-UX model, print name of	model(1)
current locale, query numeric formatting conventions of	localeconv(3C)
current page size, get	getpagesize(2)
current process, get audit ID (aid()) for	getaudit(2)
current process, set audit ID (aid()) for	setaudit(2)
current process, suspend or resume auditing on	audswitch(2)
current processes, list	ps(1)
current SCCS file editing activity, print	sact(1)
current state (X/OPEN TLI-XTI)	t_getstate(3)
current status of the UUCP system	uusnap(1M)
current system users, list	who(1)
current terminal information	cur_term(3X)
current terminal, get verbose description of	longname(3X)
current user context, get and set	getcontext(2)
current user id, print or display	whoami(1)
current user, find the slot in the utmp() file of the	ttyslot(3C)
current users and processes, list	whodo(1M)
current value of system-wide clock, get	getclock(3C)
current window	curscr(3X)
current working directory	glossary(9)
current working directory, get path-name of	getcwd(3C)
current working directory, get pathname of	getwd(3C)
curscr() - current window	curscr(3X)
Curses session, suspend	endwin(3X)
curses window, scroll	scroll(3X)
curses_intro - introduction to curses	curses_intro(3X)
cursor and window coordinates, get additional	getbegyx(3X)
cursor and window coordinates, get	getyx(3X)
cursor to end of window, clear	clrtoeol(3X)
cursor, clear from it to end of line	clrtoeol(3X)
cursor, output movement commands to the terminal	mvcur(3X)
cursor, set the cursor mode	curs_set(3X)
cursor, window location functions	move(3X)
curs_set() - set the cursor mode	curs_set(3X)
cur_term() - current terminal information	cur_term(3X)
cuserid() - get character-string login name of the user	cuserid(3S)
cut - cut out selected fields of each line in a file	cut(1)
Cyclical Redundancy Check on a file	sum(1)
c_colwidth(), C_COLWIDTH() - test for valid first byte in 16-bit character	nl_tools_16(3X)
daemon debug utility used by DDFA software, outbound connection	ocdebug(1M)
daemon for modifying Network Information Service passwd database	yppasswdd(1M)
daemon	glossary(9)
daemon that responds to SNMP requests	snmpd(1M)
daemon to maintain the nis+ password table in sync with the nis+ trusted table	ttsyncd(1M)
daemon used by DDFA software, outbound connection	ocd(1M)
daemon, gateway routing	gated(1M)

Description	Entry Name(Section)
daemon, Internet services	inetd(1M)
daemon, IP multicast routing	mrouted(1M)
daemon, kills the sendmail daemon	killsm(1M)
daemon, line printer daemon for LP requests from remote systems	rlpdaemon(1M)
daemon, network lock	lockd(1M)
daemon, NIS+ password update daemon	rpc.nispasswd(1M)
daemon, NIS+ service	rpc.nisd(1M)
daemon, password and group hashing and caching	pwgrd(1M)
daemon, PFS	pfsd(1M)
daemon, point to point protocol	cp(1)
daemon, system physical environment	envd(1M)
daemon, timed-job execution	cron(1M)
daemon, Uninterruptible Power System (UPS) monitor	ups_mond(1M)
daemon: audit-overflow monitor daemon	audomon(1M)
daemons, NFS	nfsd(1M)
daily accounting shell procedure	runacct(1M)
daily system activity report package	sal(1M)
damaged file system, patch up (generic)	fsdb(1M)
damaged HFS file system, patch up	fsdb_hfs(1M)
data allocation space of object files, print section sizes and	size(1)
data and stack space, allocate then lock process into memory	datalock(3C)
data base compiler, terminfo	tic(1M)
data base of terminal-type for each tty port	ttysize(4)
data base, terminfo, de-compile	untic(1M)
Data Communications and Terminal Controller Device File Access software	ddfa(7)
data encryption	glossary(9)
data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)
data from a file, read	read(2)
data or expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
data order for display/keyboard, convert file	forder(1)
data order, convert string	strord(3C)
data over connection, receive (X/OPEN TLI-XTI)	t_rcv(3)
data path width (in bits), set	io_width_ctl(3I)
data pointer for binary search tree, get	tsearch(3C)
data returned by stat/fstat/lstat system call	stat(5)
data segment and shared memory, attach or detach	shmop(2)
data segment space allocation, change	brk(2)
data to a file, write	write(2)
data to check the network, scatter	spray(3N)
data transfer rate, inform system of required minimum I/O	io_speed_ctl(3I)
data translation of ELF files	elf_xlate(3E)
data types, basic integer	inttypes(5)
data types, system primitives	types(5)
data unit from remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
data unit, send to transport user (X/OPEN TLI-XTI)	t_sndudata(3)
data, expand or compress	compress(1)
data, get character or word from a stream file	getc(3S)
data, get wide character from a stream file	getwc(3C)
data, lock in memory	plock(2)
data: manipulate crash dump data	crashutil(1M)
database access functions, NIS+	nis_db(3N)
database and directory structure, Network Information Service	ypfiles(4)
database converter, DHCP client	dhcplib2conf(1M)
database entry, manipulate device assignment	getdvagent(3)
database entry, manipulate protected password	getprpwent(3)
database entry, manipulate system default	getprdfent(3)
database entry, manipulate terminal control	getprtcent(3)
database file, trusted system device assignment	devassign(4)
database files and directory structure, NIS+	nisfiles(4)
database for public keys	publickey(4)
database for the mail aliases file, rebuild	newaliases(1M)

Index All Volumes

Description	Entry Name(Section)
database from NIS server to local node, transfer NIS	ypxfr(1M)
database maps for sendmail, creating	makemap(1M)
database subroutines (new multiple database version)	ndbm(3X)
database subroutines (old version - see also ndbm(3X))	dbm(3X)
database, check internal consistency of Authentication	authck(1M)
database, create or rebuild Network Information Service	ypmake(1M)
database, daemon for modifying Network Information Service passwd	yppasswdd(1M)
database, force propagation of a Network Information Service	yppush(1M)
database, host names	hosts(4)
database, make a Network Information System	makedbm(1M)
database, network configuration	netconfig(4)
database, network name	networks(4)
database, pathalias, access and manage the	uupath(1)
database, protocol name	protocols(4)
database, relational, join two relations in	join(1)
database, RPC program number	rpc(4)
database, service name	services(4)
database, termcap , emulation	tgetent(3X)
databases, build and install Network Information Service	ypinit(1M)
databases, security	authcap(4)
datacomm line speed and terminal settings used by getty	gettydefs(4)
datagram protocol, Internet user	UDP(7P)
datalock() - lock process into memory after allocating data and stack space	datalock(3C)
date and ctime , time zone adjustment table for	tztab(4)
date and time, convert to string	ctime(3C)
date and time, convert to string	strftime(3C)
date and time, convert to wide-character string	wcsftime(3C)
date and time, convert user format	getdate(3C)
date and time, get more precisely (Version 7 compatibility only)	ftime(2)
date and time, set via NTP	ntpdate(1M)
date - display or set the system-clock date and time	date(1)
date, display current or set to new value	date(1)
date, get system clock	gettimeofday(2)
date, set	stime(2)
daylight() - Daylight Savings Time flag	ctime(3C)
dbmclose() - close currently open database (old single-data-base version)	dbm(3X)
dbmopen() - open a single database (old single-data-base version)	dbm(3X)
dbm_clearerr() - reset error condition on named database	ndbm(3X)
dbm_close() - close an open database	ndbm(3X)
dbm_delete() - delete a database key and associated contents	ndbm(3X)
dbm_error() - error in reading or writing in a database	ndbm(3X)
dbm_fetch() - access a database entry under a key	ndbm(3X)
dbm_firstkey() - get first key in a database	ndbm(3X)
dbm_nextkey() - get next key in a database	ndbm(3X)
dbm_open() - open a database for access	ndbm(3X)
dbm_store() - store an entry under a key in a database	ndbm(3X)
db_add_entry() - NIS+ database access functions	nis_db(3N)
db_checkpoint() - NIS+ database access functions	nis_db(3N)
db_create_table() - NIS+ database access functions	nis_db(3N)
db_destroy_table() - NIS+ database access functions	nis_db(3N)
db_first_entry() - NIS+ database access functions	nis_db(3N)
db_free_result() - NIS+ database access functions	nis_db(3N)
db_initialize() - NIS+ database access functions	nis_db(3N)
db_list_entries() - NIS+ database access functions	nis_db(3N)
db_next_entry() - NIS+ database access functions	nis_db(3N)
db_remove_entry() - NIS+ database access functions	nis_db(3N)
db_reset_next_entry() - NIS+ database access functions	nis_db(3N)
db_standby() - NIS+ database access functions	nis_db(3N)
db_table_exists() - NIS+ database access functions	nis_db(3N)
db_unload_table() - NIS+ database access functions	nis_db(3N)
dc - desk calculator	dc(1)

Description	Entry Name(Section)
dcc - control SCSI disk array read/write caching	dcc(1M)
dcopy - copy HFS file system with compaction	dcopy(1M)
dd - convert, reblock, translate, and copy a (tape) file	dd(1)
ddfa - Data Communications and Terminal Controller Device File Access software	ddfa(7)
DDFA software and Telnet port identification feature, dedicated ports file used by	dp(4)
DDFA software, configuration file, used by	pcf(4)
DDFA software, dedicated ports parser used by	dpp(1M)
DDFA software, outbound connection daemon used by	ocd(1M)
DDS tape cartridge, partition	mediainit(1)
de-compile terminfo data base	untic(1M)
dead.letter file	sendmail(1M)
debug file system (generic)	fsdb(1M)
debug HFS file system	fsdb_hfs(1M)
debug utility used by DDFA software, outbound connection daemon	ocdebug(1M)
debugger, VxFS file system	fsdb_vxfs(1M)
debugger: absolute debugger	adb(1)
debugger: assembler debugger	adb(1)
debugger: object code debugger	adb(1)
debugging tool, STREAMS debugging tool	strdb(1M)
decimal ASCII string, convert long integer to	ltostr(3C)
decimal equivalents: ASCII character set	ascii(5)
decimal library, packed, HP 3000-mode	hppac(3X)
decode a file encoded by uuencode	uuencode(1)
decode/encode files	crypt(1)
decompose floating-point number	modf(3M)
decrease physical extents allocated to LVM logical volume	lvreduce(1M)
decrypt and store secret key	keylogin(1)
decrypt/encrypt files	crypt(1)
dedicated line, reserve for a purpose	ripoffline(3X)
dedicated ports file used by DDFA software and Telnet port identification feature	dp(4)
dedicated ports parser used by DDFA software	dpp(1M)
default database entry, manipulate system	getprdfent(3)
default database file, trusted system	default(4)
default disk section sizes, calculate	disksecn(1M)
default kernel files, update	kmupdate(1M)
default - label default in switch statement	csh(1)
default login shell, change	chsh(1)
default message catalog, set	setcat(3)
default search path	glossary(9)
default stacksize, change	pthread_default_stacksize_np(3T)
default - system default database file for a trusted system	default(4)
default values: display NIS+ default values	nisdefaults(1)
default window	stdscr(3X)
default_disk_ir tunable parameter	mount(1M)
define additional severities	addsev(3C)
define additional signal stack space	sigspace(2)
define and describe audit system events	audeventstab(4)
define I/O device interrupt (fault) conditions	io_on_interrupt(3I)
define kernel process accounting output file or disable accounting	acct(1M)
define label for formatting routines	setlabel(3)
define what to do upon receipt of a signal	signal(2)
define window coordinate transformation	mvderwin(3X)
definition, display system	sysdef(1M)
definitions, memory mapping	mman(5)
definitions, regular expression and pattern matching notation	regexp(5)
def_prog_mode() - save or restore program or shell terminal modes	def_prog_mode(3X)
def_shell_mode() - save terminal modes as the "shell" state	def_prog_mode(3X)
degree-valued arccosine functions	acosd(3M)
degree-valued arcsine functions	asind(3M)
degree-valued arctangent functions	atand(3M)
degree-valued arctangent-and-quadrant functions	atan2d(3M)

Index All Volumes

Description	Entry Name(Section)
degree-valued cosine functions	cosd(3M)
degree-valued sine functions	sind(3M)
degree-valued tangent functions	tand(3M)
delay and insert capability, for terminal	has_ic(3X)
delay command execution	at(1)
delay mode, control input character delay mode	halfdelay(3X)
delay_output() - delay output	delay_output(3X)
delch() , mvdelch() , mvwdelch() , wdelch() - delete character from a window	delch(3X)
delete a directory	rmdir(1)
delete a file or directory	rm(1)
delete a group from the system	groupdel(1M)
delete a node from a binary search tree	tsearch(3C)
delete a user login from the system	userdel(1M)
delete a window	delwin(3X)
delete allocated signal stack space	sigspace(2)
delete and modify user credentials for an authentication service	pam_setcred(3)
delete DOS files or directories	dosrm(1)
delete file	unlink(2)
delete NIS+ directories	nismkdir(1)
delete NIS+ objects from the namespace	nismrm(1)
delete or insert lines into a window	insdelln(3X)
delete secret key stored with keyserv	keylogout(1)
delete selected characters	tr(1)
delete user crontab file	crontab(1)
delete() - delete key and data under it (old single-data-base version)	dbm(3X)
delete, add, or modify delete access control list entry	setaclentry(3C)
delete, add, update a kernel module	kminstall(1M)
delete, copy, add, modify, or summarize file access control lists (ACLs)	chacl(1)
delete-character features, hardware, enable or disable use of	idcok(3X)
deleteln() , wdeleteln() - delete lines in window	deleteln(3X)
deletes print objects	pdelete(1)
delscreen() - free storage associated with a screen	delscreen(3X)
delta (change) to an SCCS file, make a	delta(1)
delta commentary of an SCCS delta, change	cdc(1)
delta from an SCCS file, remove a	rmDEL(1)
delta	glossary(9)
delta - make a delta (change) to an SCCS file	delta(1)
delwin() - delete a window	delwin(3X)
del_curterm() , restartterm() , set_curterm() , setupterm() - interfaces to terminfo database	del_curterm(3X)
demon	glossary(9)
deny or permit <i>write(1)</i> messages from other users to terminal	mesg(1)
dependencies, list dynamic, of executable files or shared libraries	ldd(1)
depot, modify software products target in root or	swmodify(1M)
depots and roots, register or unregister	swreg(1M)
deroff - remove nroff, tbl, and neqn constructs	deroff(1)
derwin() - relative window creation function	derwin(3X)
DES encryption key, generate a	makekey(1)
descend a directory hierarchy recursively, executing a function	ftw(3C)
describe audio file	attributes(1)
describe audit system events, define and	audeventstab(4)
describe characteristics of a disk device	diskinfo(1M)
description file for at , batch , and crontab queues	queuedefs(4)
description of common HP-UX terms	glossary(9)
description of disk by its name, get	getdiskbyname(3C)
description of HP-UX header file organization	stdsyms(5)
description of supported languages	lang(5)
description, DOS Interchange Format	dosif(4)
description, host name resolution	hostname(5)
description, logical interchange format	lif(4)
description, secure internet services	sis(5)

Description	Entry Name(Section)
description, verbose, of current terminal	longname(3X)
descriptor file entry, get file system (BSD 4.2 compatibility only)	getfsent(3X)
descriptor, close a file	close(2)
descriptor, map stream pointer to file	fileno(3S)
descriptor: update an ELF descriptor	elf_update(3E)
desk calculator	dc(1)
destroy a mutex attribute object	pthread_mutexattr_init(3T)
destroy a mutex	pthread_mutex_init(3T)
destroy a read-write lock attribute object	pthread_rwlockattr_init(3T)
destroy a read-write lock	pthread_rwlock_init(3T)
destroy a thread attribute object	pthread_attr_dinit(3T)
destroy a thread-specific data key	pthread_key_create(3T)
destroy an unnamed semaphore	sem_destroy(2)
destroy mass storage data for security purposes (use -r option)	mediainit(1)
destroy or initialize a thread condition variable	pthread_cond_init(3T)
destroy or initialize a thread condition variable attributes object	pthread_condattr_init(3T)
detach a name from a STREAMS-based file descriptor	fdetach(3C)
detach a STREAMS-based file descriptor	fdetach(1M)
detach a thread to reclaim its resources when it terminates	pthread_detach(3T)
detach shared memory from data segment	shmop(2)
determine accessibility of a file	access(2)
determine current signal stack space	sigspace(2)
determine file type	file(1)
determine file type for ELF files	elf_kind(3E)
determine how last I/O read terminated	io_get_term_reason(3I)
determine whether a screen has been refreshed	isendwin(3X)
determine who is logged in on local network machines	rusers(1)
determined processor IDs	pthread_processor_bind_np(3T)
devassign - trusted system device assignment database file	devassign(4)
device (special) file, list an I/O	lssf(1M)
device (special) file, make	mksf(1M)
device (special) file, remove a	rmsf(1M)
device (special) files, install	insf(1M)
device access drivers, cartridge tape	ct(7)
device address	glossary(9)
device and FIFO files, create	mknod(1M)
device assignment database entry, manipulate	getdvagent(3)
device configuration and status, display LAN0	lanscan(1M)
device control commands, generic I/O	ioctl(5)
device driver, flexible or "floppy" disk	floppy(7)
device driver, PS/2 keyboard and mouse	ps2(7)
device driver, SCSI direct access	scsi_disk(7)
device driver, SCSI media changer	autochanger(7)
device driver, SCSI sequential access	scsi_tape(7)
device driver: STREAMS pass through driver	clone(7)
device drivers	(see special files)
device drivers in the system, list	lsdev(1M)
device drivers, Small Computer System Interface (SCSI)	scsi(7)
Device File Access software, Data Communications and Terminal Controller	ddfa(7)
device file	glossary(9)
device file, block mode terminal	blmode(7)
device file, FIFO, make a	mkfifo(3C)
device files, network file system	nfs(7)
device for interleaved paging and swapping, add swap	swapon(2)
device for paging, enable	swapon(1M)
device	glossary(9)
device I/O interrupt (fault) control	io_on_interrupt(3I)
device ID to file path, map	devnm(3)
device name	devnm(1M)
device numbers, header file of macros for handling	mknod(5)
device special file, control character	ioctl(2)

Index

All Volumes

Description	Entry Name(Section)
device special files, introduction	intro(7)
device, control a SCSI	scsictl(1M)
device, control terminal (Bell Version 6 compatibility)	stty(2)
device, describe characteristics of a disk	diskinfo(1M)
device, who is currently using	fuser(1M)
device: STREAMS device	isastream(3C)
device; update, install, or remove boot programs from a disk	mkboot(1M)
devices - file of driver information for insf , mksf , lssf	devices(4)
devnm - device name	devnm(1M)
devnm() - map device ID to file path	devnm(3)
df - report number of free CDFS, HFS, or NFS file system disk blocks	df_hfs(1M)
df - report number of free disk blocks on a VxFS file system	df_vxfs(1M)
df - report number of free file system disk blocks (generic)	df(1M)
df_vxfs - report number of free disk blocks on a VxFS file system	df_vxfs(1M)
DHCP client database converter	dhcplib2conf(1M)
DHCP elements of bootpd, command line tool	dhcptools(1M)
DHCP support script, system configuration	auto_parms(1M)
dhcplib2conf - DHCP client database converter	dhcplib2conf(1M)
dhcptools - command line tool for DHCP elements of bootpd	dhcptools(1M)
diag0 - diagnostic interface to I/O subsystem	diag0(7)
diagnostic information, dynamic linking process	dlerror(3C)
diagnostic interface to I/O subsystem	diag0(7)
diagnostic messages, collect to form system error log	dmesg(1M)
diagnostics	fcmsutil(1M)
diagnostics	fcutil(1M)
diagnostics, local area network	lanadmin(1M)
dial() - establish an out-bound terminal line connection	dial(3C)
dialer description file format, ppp	ppp.Dialers(4)
dialups, d_passwd - dialup security control	dialups(4)
dialups file	login(1)
diff, diffh - differential file comparator	diff(1)
diff between sorted files; reject/select common lines	comm(1)
diff file to an original, applying	patch(1)
diff, big	bdiff(1)
diff3 - three-way differential file comparison	diff3(1)
difference function, positive	fdim(3M)
differences among three files	diff3(1)
differences between files, show side-by-side	sdiff(1)
differences between large files, find	bdiff(1)
differences between two files	diff(1)
differences between two files, mark	diffmk(1)
differential file comparator	diff(1)
diffmk - mark differences between files	diffmk(1)
difftime() - difference between two calendar time values	ctime(3C)
dir - format of directories	dir(4)
dircmp - directory comparison	dircmp(1)
direct access device driver, SCSI	scsi_disk(7)
direct disk device access drivers	disk(7)
directories to export to NFS clients	exports(4)
directories to export to PFS clients	pfs_exports(5)
directories to NFS clients, export and unexport	exportfs(1M)
directories, binary, install object files in	cpset(1M)
directories, export or unexport to PFS clients	pfs_exportfs(1M)
directories: create, remove directories in a path	mkdirp(3G)
directories: creating NIS+ directories	nismkdir(1)
directories: list the contents	nisl(1)
directory and file structures, statd	sm(4)
directory clean-up, uuCP spool	uucleanup(1M)
directory entries and directory streams, format of	dirent(5)
directory file, make	mknod(2)
directory file, remove a	rmdir(2)

Description	Entry Name(Section)
directory format	dir(4)
directory	glossary(9)
directory hierarchy, recursively descend a, executing a function	ftw(3C)
directory name, print working	pwd(1)
directory object, NIS+, update the public keys	nisupdkeys(1M)
directory pointer array, sort a	scandir(3C)
directory service, Internet user name	whois(1)
directory stream, directory and associated, open for access	directory(3C)
directory streams and directory entries, format of	dirent(5)
directory streams, HP-UX, format of	ndir(5)
directory structure and database files, NIS+	nisfiles(4)
directory structure, Network Information Service database and	ypfiles(4)
directory, change working	cd(1)
directory, get entries in a file-system-independent format	getdirentries(2)
directory, get pathname of current working	getwd(3C)
directory, move directory subtree and files to another directory	mv(1)
directory, move multiple files to another directory	mv(1)
directory, rename directory	mv(1)
directory, scan a	scandir(3C)
directory, symbolic links between directories, create	ln(1)
directory: access, open, or close a directory and associated directory stream	directory(3C)
directory: change root directory	chroot(2)
directory: change root directory for a command	chroot(1M)
directory: change working directory	chdir(2)
directory: compare contents of two directories	dircmp(1)
directory: compress files in a directory	compress(1)
directory: copy directory subtree and files to another directory	cp(1)
directory: copy multiple files to a directory	cp(1)
directory: create a directory	mkdir(1)
directory: create a DOS directory	dosmkdir(1)
directory: expand compressed files in a directory	compress(1)
directory: get path-name of current working directory	getcwd(3C)
directory: list contents of directory	ls(1)
directory: list contents of DOS directories	dosls(1)
directory: make a directory file	mkdir(2)
directory: move a directory (requires super-user)	mvdir(1M)
directory: print working directory name	pwd(1)
directory: read portable archive	pax(1)
directory: remove directory	rmdir(1)
directory: remove DOS files or directories	dosrm(1)
directory: remove entry	unlink(2)
directory: scan a directory	scandir(3C)
directory: search directory tree for files	find(1)
directory: search for named file in named directories	pathfind(3G)
directory: write portable archive	pax(1)
dirent.h - format of directory streams and directory entries	dirent(5)
dirname, basename - extract portions of path names	basename(1)
dirname() - return path name of parent directory	basename(3C)
dirs - print the directory stack	csh(1)
disable accounting or define kernel process accounting output file	acct(1M)
disable - disable LP printers	enable(1)
disable or enable abbreviation of function keys	keypad(3X)
disable or enable flush on interrupt	intrflush(3X)
disable or enable I/O interrupts for the associated eid()	io_interrupt_ctl(3I)
disable or enable process accounting	acct(2)
disable use of certain terminal capabilities	filter(3X)
disable/enable block during read	nodelay(3X)
disable/enable immediate terminal refresh	immedok(3X)
disable/enable meta-keys	meta(3X)
disable/enable newline translation	nl(3X)
disable/enable queue flushing	noqiflush(3X)

Index All Volumes

Description	Entry Name(Section)
disable/enable terminal echo	echo(3X)
disable/enable use of hardware insert- and delete-character features	idcok(3X)
discard file (bit bucket)	null(7)
discard input	flushinp(3X)
disconnect information (X/OPEN TLI-XTI)	t_rcvdis(3)
disconnect request, send user-initiated (X/OPEN TLI-XTI)	t_snddis(3)
disk accounting data of VxFS file system , disk usage by user ID	diskusg(1M)
disk accounting data, disk usage by user ID	diskusg(1M)
disk accounting, perform	acctsh(1M)
disk array, configuration table	arraytab(4)
disk array, configure a logical unit (LUN)	cfl(1M)
disk array, configuring	newarray(1M)
disk array, control read/write caching	dcc(1M)
disk array, describe characteristics of	arrayinfo(1M)
disk array, display status of	dsp(1M)
disk array, download firmware to	dlf(1M)
disk array, formatting a LUN for	format(1M)
disk array, repair LUN parity information	rpr(1M)
disk array, scan LUNs for parity consistency	pscan(1M)
disk array, scan LUNs for parity consistency	scn(1M)
disk array, search system for	arrayscan(1M)
disk array, set spindle synchronization state of drives in	sss(1M)
disk array: set physical drive parameters for	spd(1M)
disk blocks (generic), report number of free file system	df(1M)
disk blocks on a VxFS file system	df_vxfs(1M)
disk blocks, report number of free (Berkeley version)	bdf(1M)
disk blocks, report number of free, CDFS, HFS, or NFS file system	df_hfs(1M)
disk clusters, DOS, report number of free	dosdf(1)
disk description by its name, get	getdiskbyname(3C)
disk description file	disktab(4)
disk device driver, flexible or "floppy"	floppy(7)
disk device, describe characteristics of a	diskinfo(1M)
disk device, direct access drivers	disk(7)
disk device; update, install, or remove boot programs from a	mkboot(1M)
disk - direct disk device access drivers	disk(7)
disk directory format	dir(4)
disk layout of VxFS file system, upgrade	vxupgrade(1M)
disk quota status of specified file system, determine	fsclean(1M)
disk quotas	quota(5)
disk quotas, manipulate	quotactl(2)
disk quotas, summarize for a file system	repquota(1M)
disk section sizes, calculate default	disksecn(1M)
disk space management	lvm(7)
disk space, DOS, report amount of available	dosdf(1)
disk space, recover	freedisk(1M)
disk storage space, preallocate	prealloc(1)
disk storage, preallocate fast	prealloc(2)
disk usage accounting records, create	acct(1M)
disk usage and limits, display	quota(1)
disk usage by login name, compute	acct(1M)
disk usage, generate disk accounting data by user ID	diskusg(1M)
disk usage, generate disk accounting data of VxFS file system by user ID	vxdiskusg(1M)
disk usage, summarize	du(1)
disk, flush buffers to	sync(2)
disk, flush unwritten system buffers to	sync(1M)
disk, SCSI direct access device driver	scsi_disk(7)
disk, synchronize a file's in-core state with its state on	fsync(2)
disk: periodically flush unwritten system buffers to disk	syncer(1M)
diskinfo - describe characteristics of a disk device	diskinfo(1M)
diskless (remote) boot server	rbootd(1M)
diskless client configuration information file	info(4)

Description	Entry Name(Section)
disksecn - calculate default disk section sizes	disksecn(1M)
disktab - disk description file	disktab(4)
diskusg - generate disk accounting data by user ID	diskusg(1M)
dismount (unmount) a file system (generic)	mount(1M)
dismount (unmount) CDFS file systems	mount_cdfs(1M)
dismount (unmount) HFS file systems	mount_hfs(1M)
dismount (unmount) NFS file systems	mount_nfs(1M)
display (print) arguments	echo(1)
display (print) formatted arguments	printf(1)
display address resolution	arp(1M)
display and modify variables in the stable storage	setboot(1M)
display and update information about top processes on system	top(1)
display audit information as requested by parameters	audisp(1M)
display call graph execution profile data	gprof(1)
display contents of the NIS+ transaction log	nislog(1M)
display current system-clock date and time	date(1)
display disk usage and limits	quota(1)
display el m user and system aliases	elmalias(1)
display file on soft-copy terminals	pg(1)
display file on CRT terminals	more(1)
display information about computer system	uname(1)
display information about LVM logical volumes	lvdisplay(1M)
display information about LVM volume groups	vgdisplay(1M)
display information about physical volumes in LVM volume group	pvdisplay(1M)
display job information and remove jobs	swjob(1M)
display LAN device configuration and status	lanscan(1M)
display message in standard format	pfmt(3C)
display monitor profile data	prof(1)
display name of current host system	hostname(1)
display Network Information Service domain name	domainname(1)
display network status	netstat(1)
display NIS+ default values	nisdefaults(1)
display NIS+ error messages	niserror(1)
display NIS+ error messages	nis_error(3N)
display NIS+ tables and objects	niscat(1)
display or change event or system call audit status	audevent(1M)
display or set audit file information	audsys(1M)
display STREAMS structures	strdb(1M)
display system and user login data	logins(1M)
display system definition	sysdef(1M)
display user login data	listusers(1M)
display /keyboard data order, convert file	forder(1)
displayed CRT terminal output, record, simultaneously in a file	script(1)
displays formatted message on standard error and console	fmtmsg(3C)
displays the last part of the mail log	mtail(1M)
distance function, Euclidean (hypotenuse)	hypot(3M)
distribute , install, monitor, create, and manage software	sd(5)
Distributed Print Service, creates print objects	pdcreate(1)
Distributed Print Service, enables printers to accept jobs and logs to log	pdenable(1)
Distributed Print Service, lists selected attribute values	pdl(1)
Distributed Print Service, removes all jobs from the specified object	pdclean(1)
Distributed Print Service, stops printers from accepting jobs and logs from logging	pddisable(1)
div (), ldiv () - integer division and remainder	div(3C)
divide mirrored LVM logical volume into two logical volumes	lvsplit(1M)
division and remainder, integer	div(3C)
divpage - divide pages for two-sided printing	lpfilter(1)
dlclose () - close a shared object	dlclose(3C)
dld.sl - dynamic loader	dld.sl(32)
dld.sl - dynamic loader	dld.sl(5)
dld.sl - dynamic loader	dld.sl(64)
dlerror () - get diagnostic information from dynamic linking process	dlerror(3C)

Index

All Volumes

Description	Entry Name(Section)
d1f - download firmware to an HP SCSI disk array	d1f(1M)
d1get() - retrieve information on loaded module (program or shared library)	d1get(3C)
d1getname() - retrieve name of load module	d1getname(3C)
d1modinfo() - retrieve information on loaded module (program or shared library)	d1modinfo(3C)
dlopen() - open a shared object	dlopen(3C)
dlsym() - get address of symbol in shared object	dlsym(3C)
dmesg - collect system diagnostic messages to form error log	dmesg(1M)
dmpxlt - dump iconv translation tables to a readable format	dmpxlt(1)
dn_comp() - resolver routines	resolver(3N)
dn_expand() - resolver routines	resolver(3N)
do nothing and return zero or non-zero exit status	true(1)
documentation, list of orderable HP-UX	manuals(5)
documents, format and print using the mm macros	mm(1)
documents, MM macro package for formatting	mm(5)
dodisk - perform disk accounting	acctsh(1M)
domain name server, Internet	named(1M)
domain name server, send signals to	sig_named(1M)
domain name, set or display Network Information Service	domainname(1)
domain protocol, local communication	UNIX(7P)
domain, get or set name of current NIS	getdomainname(2)
domain, initialize a NIS+ domain	nissetup(1M)
domainname - set or display NIS domain name	domainname(1)
DOS files: change attributes of a DOS file	doschmod(1)
DOS files: convert DOS file to HP-UX ASCII format	dos2ux(1)
DOS files: convert HP-UX ASCII file to DOS format	dos2ux(1)
DOS files: copy to or from	doscp(1)
DOS files: create a DOS directory	dosmkdir(1)
DOS files: list contents of DOS directories	dosls(1)
DOS files: remove DOS files or directories	dosrm(1)
DOS files: report number of free DOS disk clusters	dosdf(1)
dos2ux, ux2dos - convert ASCII file format	dos2ux(1)
doschmod - change attributes of a DOS file	doschmod(1)
doscp - copy to or from DOS files	doscp(1)
dosdf - report number of free DOS disk clusters	dosdf(1)
DOSIF - DOS Interchange Format description	dosif(4)
dos1l, dos1s - list contents of DOS directories	dosls(1)
dosls, dos1l - list contents of DOS directories	dosls(1)
dosmkdir - make a DOS directory	dosmkdir(1)
dosrm, dosrmdir - remove DOS files or directories	dosrm(1)
dosrmdir, dosrm - remove DOS files or directories	dosrm(1)
dot	glossary(9)
dot-dot	glossary(9)
dot-oh file	glossary(9)
dot-oh format	glossary(9)
dot-oh	glossary(9)
double-precision number, convert string to	strtod(3C)
double-precision number, convert string to long	strtold(3C)
double-precision number, convert wide character string to	wctod(3C)
doudate(), refresh(), wnoutrefresh(), wrefresh() - refresh windows and lines	doudate(3X)
downshifting	glossary(9)
dp - dedicated ports file used by DDFA software and Telnet port identification feature	dp(4)
dpp - dedicated ports parser used by DDFA software	dpp(1M)
drand48(), erand48() - generate double-precision pseudo-random numbers	drand48(3C)
draw borders from complex characters and renditions	border_set(3X)
draw borders from complex characters and renditions	box_set(3X)
draw borders from single-byte characters and renditions	border(3X)
draw borders from single-byte characters and renditions	box(3X)
draw lines from complex characters and renditions	hline_set(3X)
draw lines from single-byte characters and renditions	hline(3X)
driver	(see also special files)
driver information file for insf, mksf, lssf	devices(4)

Description	Entry Name(Section)
driver, block mode terminal	blmode(7)
driver, flexible or "floppy" disk device	floppy(7)
driver, IP network tunnel driver	tun(4)
driver, PS/2 keyboard and mouse devices	ps2(7)
driver, SCSI device control device	scsi_pt(7)
driver, SCSI direct access device	scsi_disk(7)
driver, SCSI media changer device	autochanger(7)
driver, SCSI pass-through device	scsi_ctl(7)
driver, SCSI sequential access device	scsi_tape(7)
driver: STREAMS Administrative Driver	sad(7)
driver: STREAMS log driver	strlog(7)
drivers in the system, list device	lsdev(1M)
drivers, Small Computer System Interface (SCSI) device	scsi(7)
drivers: cartridge tape device access	ct(7)
drivers: direct disk device access drivers	disk(7)
drivers: HP-HIL cooked keyboard	hilkbd(7)
drivers: HP-HIL device driver	hil(7)
drivers: raster frame-buffer display device access	framebuf(7)
dsp - display status of an HP SCSI disk array	dsp(1M)
du - summarize disk usage	du(1)
dump and restore protocol module, remote magnetic tape	rmt(1M)
dump file in octal or hexadecimal format	od(1)
dump file system information	dumpfs(1M)
dump iconv translation tables to a readable format	dmpxlt(1)
dump - incremental file system dump (for backups)	dump(1M)
dump information contained in object files	elfdump(1)
dump unwritten system buffers to disk	sync(1M)
dump unwritten system buffers to disk periodically	syncer(1M)
dump volume, prepare LVM logical volume to be	lvlnboot(1M)
dump volume, remove LVM logical volume link	lvrmboot(1M)
dump window to and reload window from a file	getwin(3X)
dump, crash dump access library	libcrash(5)
dump, incremental file system, across network	vxdump(1M)
dump, incremental file system, local	vxdump(1M)
dumpfs - dump file system information	dumpfs(1M)
dumpmsg - create message catalog file for modification	findmsg(1)
dup() - duplicate an open file descriptor	dup(2)
dup2() - duplicate an open file descriptor to a specific slot	dup2(2)
duplicate a window	dupwin(3X)
duplicate an open file descriptor	dup(2)
duplicate an open file descriptor to a specific slot	dup2(2)
duplicate entries in a table, eliminate	lsearch(3C)
dupwin() - duplicate a window	dupwin(3X)
dynamic dependencies of executable files, list	ldd(1)
dynamic file system swapping	swapon(2)
Dynamic Host Configuration Protocol (DHCP)	dhcplib2conf(1M)
dynamic linking process, diagnostic information	dlerror(3C)
dynamic loader	dld.sl(64)
dynamic loader	glossary(9)
dynamic loader, binding behavior	ld(1)
dynamically loadable kernel modules, change global search path	modpath(2)
dynamically loaded kernel module, get information	modstat(2)
d_passwd - dialup security control	dialups(4)
e - extended line-oriented text editor	ex(1)
echo - echo (print) arguments	csh(1)
echo - echo (print) arguments	echo(1)
echo - echo (print) arguments	ksh(1)
echo - echo (print) arguments	sh-bourne(1)
echo - echo (print) arguments	sh-posix(1)
echo packets	ping(1M)
Echo Request packet, send Fibre Channel Light Weight Protocol	pong(1M)

Description	Entry Name(Section)
echo single-byte character and rendition to a window and refresh	echochar(3X)
echo() - enable/disable terminal echo	echo(3X)
echo, suppress while reading password from terminal	getpass(3C)
echochar() - echo single-byte character and rendition to a window and refresh	echochar(3X)
ECHO_REQUEST packets	ping(1M)
echo_wchar() - write a complex character and immediately refresh the window	echo_wchar(3X)
ecvt(), fcvt() - convert floating-point number to string	ecvt(3C)
ed - line-oriented text editor	ed(1)
edata() - first address beyond initialized program data region	end(3C)
edit - beginner's line-oriented text editor	ex(1)
edit the password file using vi editor	vipw(1M)
edit user quotas	edquota(1M)
editing activity, print current SCCS file	sact(1)
editor and command history for interactive programs input	ied(1)
editor, audio	see audio(5)
editor: extended line-oriented text editor	ex(1)
editor: extended screen-oriented text editor	vi(1)
editor: line-oriented	ed(1)
editor: streaming (non-interactive) text editor	sed(1)
edquota - edit user quotas	edquota(1M)
effective access rights to a file, get a user's	getaccess(2)
effective and real user IDs, set	setreuid(2)
effective current user id, print or display	whoami(1)
effective group ID	glossary(9)
effective or real user or group ID, get	getuid(2)
effective user ID	glossary(9)
effective, real, and/or saved user or group IDs, set	setresuid(2)
egrep - search a file for a pattern	grep(1)
EISA configuration tool	eisa_config(1M)
eisa_config - EISA configuration tool	eisa_config(1M)
electronic address router	pathalias(1)
electronic mail, screen-oriented interface	elm(1)
element, convert floating-point number to string or string array	ecvt(3C)
ELF - executable and linking format object files	elf(3E)
ELF files, set fill byte for	elf_fill(3E)
ELF library error handling	elf_error(3E)
elf() - object file access library	elf(3E)
elf32_fsize() - return the size of an object file type for elf32 files	elf_fsize(3E)
elf32_getehdr - retrieve class-dependent object file header for ELF files	elf_getehdr(3E)
elf32_getphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf32_getshdr() - retrieve class-dependent section header for ELF files	elf_getshdr(3E)
elf32_newphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf32_xlatetof() - class-dependent data translation of ELF files	elf_xlate(3E)
elf32_xlatetom() - class-dependent data translation of ELF files	elf_xlate(3E)
elf64_fsize() - return the size of an object file type for elf64 files	elf_fsize(3E)
elf64_getphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf64_getshdr() - retrieve class-dependent section header for ELF files	elf_getshdr(3E)
elf64_newphdr() - retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
elf64_xlatetof() - class-dependent data translation of ELF files	elf_xlate(3E)
elf64_xlatetom() - class-dependent data translation of ELF files	elf_xlate(3E)
elfdump - dump information contained in object files	elfdump(1)
elf_begin() - make a file descriptor, for ELF files	elf_begin(3E)
elf_cntl() - control a file descriptor for ELF files	elf_cntl(3E)
elf_end() - finish using an ELF object file	elf_end(3E)
elf_errmsg() - ELF library error handling	elf_error(3E)
elf_errno() - ELF library error handling	elf_error(3E)
elf_fill - set fill byte for ELF files	elf_fill(3E)
elf_flagdata() - manipulate flags for ELF files	elf_flag(3E)
elf_flagehdr() - manipulate flags for ELF files	elf_flag(3E)
elf_flagelf() - manipulate flags for ELF files	elf_flag(3E)
elf_flagphdr() - manipulate flags for ELF files	elf_flag(3E)

Description	Entry Name(Section)
elf_flagscn() - manipulate flags ELF files	elf_flag(3E)
elf_flagshdr() - manipulate flags for ELF files	elf_flag(3E)
elf_getarhdr() - retrieve archive member header for ELF files	elf_getarhdr(3E)
elf_getarsym() - retrieve archive symbol table for ELF files	elf_getarsym(3E)
elf_getbase() - get the base offset for an object file	elf_getbase(3E)
elf_getdata() - manipulate section data for ELF files	elf_getdata(3E)
elf_getident() - retrieve file identification data for ELF files	elf_getident(3E)
elf_getscn() - get section information for ELF files	elf_getscn(3E)
elf_hash() - compute hash value for ELF files	elf_hash(3E)
elf_kind() - determine file type for ELF files	elf_kind(3E)
elf_ndxscn() - get section information for ELF files	elf_getscn(3E)
elf_newdata() - manipulate section data for ELF files	elf_getdata(3E)
elf_newscn() - get section information for ELF files	elf_getscn(3E)
elf_next() - provide sequential archive member access for ELF files	elf_next(3E)
elf_nextscn() - get section information for ELF files	elf_getscn(3E)
elf_rand() - random archive member access for ELF files	elf_rand(3E)
elf_rawdata() - manipulate section data for ELF files	elf_getdata(3E)
elf_rawfile() - retrieve uninterpreted file contents for ELF files	elf_rawfile(3E)
elf_strptr() - make a string pointer for ELF files	elf_strptr(3E)
elf_update() - update an ELF descriptor	elf_update(3E)
elf_version() - coordinate ELF library and application versions	elf_version(3E)
eliminate .so's from nroff input	soelim(1)
eliminate a directory	rmdir(1)
eliminate a file or directory	rm(1)
eliminate adjacent repeated lines in a file	uniq(1)
eliminate duplicate entries in a table	lsearch(3C)
eliminate multiple adjacent blank lines, reduce to single blank line	ssp(1)
elm - process electronic mail through a screen-oriented interface	elm(1)
elm user and system aliases, verify and display	elmalias(1)
elmalias - display and verify elm user and system aliases	elmalias(1)
empty administration file owned by adm with mode 664, create	acctsh(1M)
emulate /etc/termcap access routines	termcap(3X)
Emulation module, STREAMS pty	ptem(7)
emulation of termcap database	tgetent(3X)
emulator, IBM 3179G/3192G, 3270, 3777 terminal	sna(1)
emulator, terminal; call another (UNIX) system	cu(1)
enable device or file system for paging	swapon(1M)
enable - enable LP printers	enable(1)
enable or disable block during read	nodelay(3X)
enable or disable flush on interrupt	intrflush(3X)
enable or disable I/O interrupts for the associated eid()	io_interrupt_ctl(3I)
enable or disable immediate terminal refresh	immedok(3X)
enable or disable process accounting	acct(2)
enable or disable use of hardware insert- and delete-character features	idcok(3X)
enable SRQ line on HP-IB, allow interface to	hpiib_rqst_srvce(3I)
enable/disable abbreviation of function keys	keypad(3X)
enable/disable meta-keys	meta(3X)
enable/disable newline translation	nl(3X)
enable/disable queue flushing	noqiflush(3X)
enable/disable terminal echo	echo(3X)
enables paused objects to resume operation	pdresume(1)
enables printers to accept jobs and logs to log	pdenable(1)
encode a binary file for transmission by mailer	uuencode(1)
encode/decode files	crypt(1)
encrypt() - generate hashing encryption	crypt(3C)
encrypt/decrypt files	crypt(1)
encryption key, generate a DES	makekey(1)
encryption keys file format, ppp	ppp.Keys(4)
encryption keys, server for storing	keyserv(1)
encryption on large strings, generate hashing	bigcrypt(3C)
encryption, hashing, generate	crypt(3C)

Index

All Volumes

Description	Entry Name(Section)
encryption, password	crypt(3C)
encrypt_r() - generate hashing encryption	crypt(3C)
end locations of allocated regions in program	end(3C)
end network host entry	gethostent(3N)
end of line, clear from cursor to end of line	clrtoeol(3X)
end part of a file, get lines from	tail(1)
end protocol entry	getprotoent(3N)
end service entry	getservent(3N)
end - terminate foreach or while loop	csh(1)
end() - first address beyond uninitialized program data region	end(3C)
end-of-file	glossary(9)
enddvagent() - free memory and close file	getdvagent(3)
endexportent() - access exported file system information	exportent(3N)
endfsent() - close file system descriptor file	getfsent(3X)
endgrent() - close currently open group() file	getgrent(3C)
endhostent() - end network host entry	gethostent(3N)
endhostent_r() - end network host entry (thread-safe)	gethostent(3N)
endmntent() - close file system description file	getmntent(3X)
endnetconfig() - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
endnetconfig() - get network configuration data base entry	getnetconfig(3N)
endnetent() - get network entry	getnetent(3N)
endnetgrent() - get network group entry	getnetgrent(3C)
endpoint for communication, create an	socket(2)
endprdfent() - manipulate system default database entry	getprdfent(3)
endprotoent() - end protocol entry	getprotoent(3N)
endprotoent_r() - end protocol entry (thread-safe)	getprotoent(3N)
endprpwent() - manipulate protected password database entry	getprpwent(3)
endprtcent() - manipulate terminal control database entry	getprtcent(3)
endpwent() - close currently open password file	getpwent(3C)
endservent() - end service entry	getservent(3N)
endservent_r() - end service entry (thread-safe)	getservent(3N)
endspent() - close currently open secure password file	getspent(3C)
endspwent() - close currently open secure password file	getspwent(3X)
endsw - terminate switch statement	csh(1)
endusershell() - close legal user shells file	getusershell(3C)
endutent() - close currently open utmp() file	getut(3C)
endutent_r() - close currently open utmp() file	getut(3C)
endutxent() - close currently open utmpx() file	getutx(3C)
endwin() - suspend Curses session	endwin(3X)
entries from a directory, get in a file-system-independent format	getdirent(2)
entries from name list, get	nlist(3C)
entries in a table, eliminate duplicate	lsearch(3C)
entries, directory, format of	dirent(5)
entry format, ioconfig	ioconfig(4)
entry format, user accounting files btmp() , utmp() , and wtmp()	utmp(4)
entry from group() file, get	getgrent(3C)
entry from password file, get	getpwent(3C)
entry from secure password file, get	getspent(3C)
entry from secure password file, get	getspwent(3X)
entry, get file system descriptor file (BSD 4.2 compatibility only)	getfsent(3X)
entry, get RPC	getrpc(3C)
entry, get, set, or end network host	gethostent(3N)
entry, get, set, or end protocol	getprotoent(3N)
entry, get, set, or end service	getservent(3N)
entry, manipulate protected password database	getprpwent(3)
entry, manipulate system default database	getprdfent(3)
entry, manipulate terminal control database	getprtcent(3)
entry, network group, get or set	getnetgrent(3C)
entry, print out a manual; find manual information by keywords	man(1)
entry, write password file	putpwent(3C)
env - set environment for command execution	env(1)

Description	Entry Name(Section)
envd - system physical environment daemon	envd(1M)
environ - user environment variables	environ(5)
Environment (CUE), HP Character-Terminal User	cue(1)
environment daemon, system physical	envd(1M)
environment for command execution, set	env(1)
environment	glossary(9)
environment list, search for value of specified variable name	getenv(3C)
environment macros and functions, floating-point	fenv(5)
environment variable, search environment list for value of	getenv(3C)
environment variable, TZ	date(1)
environment variables, user	environ(5)
environment, change or add value to	putenv(3C)
environment, clear the process	clearenv(3C)
environment, login shell script to set up user's	profile(4)
environment, print out the	printenv(1)
environment, save/restore stack for non-local goto	setjmp(3C)
Environment, SoftBench Software Development	softbench(1)
environment: getting floating-point	fegetenv(3M)
environment: saving floating-point	feholdexcept(3M)
environment: setting floating-point	fesetenv(3M)
environment: updating floating-point	feupdateenv(3M)
EOF	glossary(9)
EOI mode for HP-IB file, control	hpib_eoi_ctl(3I)
Epoch	glossary(9)
erase and reinitialize mass storage media (use -r option)	mediainit(1)
erase character	eraseswchar(3X)
erase character, single-byte	erasechar(3X)
erase terminal screen	clear(1)
erase() - clear a window	clear(3X)
erasechar() - single-byte erase character	erasechar(3X)
eraseswchar() - current erase character	eraseswchar(3X)
erf() - error function	erf(3M)
erfc() - complementary error function	erf(3M)
errno() - error indicator for system calls	errno(2)
errno() - system error messages	perror(3C)
error function	erf(3M)
error handling, ELF library	elf_error(3E)
error indicator for system calls	errno(2)
error information from unit data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)
error log files, remove outdated STREAMS error log files	strclean(1M)
error log, collect system diagnostic messages to form	dmesg(1M)
error message file, extract error messages from C source into	mkstr(1)
error message function (X/OPEN TLI-XTI)	t_error(3)
error message string, get PAM	pam_strerror(3)
error message, produce (X/OPEN - XTI)	t_strerror(3)
error messages from the STREAMS log driver	strerr(1M)
error messages, display NIS+	nis_error(3N)
error messages, extract from C source into a file	mkstr(1)
error messages, system	perror(3C)
error messages: displaying	niserror(1)
error or warning message, libcrash, print	cr_perror(3)
error processing with t_rcvuderr(3)	t_sndudata(3)
error status, asynchronous I/O	aio_error(2)
errors, find spelling	spell(1)
errors, library routines for server side remote procedure call errors	rpc_svc_err(3N)
establish an out-bound terminal line connection	dial(3C)
establish connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
establish time limit for I/O operations	io_timeout_ctl(3I)
etext() - first address beyond program text region	end(3C)
EUC, set and get code widths for ldterm	eucset(1)
Euclidean distance (hypotenuse) function	hypot(3M)

Description	Entry Name(Section)
eucset - set and get EUC code widths for ldterm	eucset(1)
eval - read arguments as shell input and execute result	csh(1)
eval - read arguments as shell input and execute resulting commands	ksh(1)
eval - read arguments as shell input and execute resulting commands	sh-bourne(1)
eval - read arguments as shell input and execute resulting commands	sh-posix(1)
evaluate arguments as an expression	expr(1)
evaluate condition for true or false	test(1)
event on transport endpoint, look at current event (X/OPEN TLI-XTI)	t_look(3)
event or system call audit status, change or display	audevent(1M)
event trace messages to standard output, write STREAMS event trace messages	strace(1M)
events and system calls currently being audited, get	getevent(2)
events and system calls to be audited	setevent(2)
events, audit system, define and describe	audeventstab(4)
ex - extended line-oriented text editor	ex(1)
examine and change blocked signals	sigprocmask(2)
examine and change signal action	sigaction(2)
examine and change signal action	sigwait(2)
examine and change the signal mask of the calling thread	pthread_sigmask(3T)
examine pending signals	sigpending(2)
exception flags: getting floating-point	fegetexceptflag(3M)
exception flags: setting floating-point	fesetexceptflag(3M)
exception trap enable bits: getting	fegettrapenable(3M)
exception trap enable bits: setting	fesettrapenable(3M)
exceptions: clearing floating-point	feclearexcept(3M)
exceptions: raising floating-point	feraiseexcept(3M)
exceptions: testing floating-point	fetestexcept(3M)
exchange, portable archive	pax(1)
exec - execute command without creating new process	csh(1)
exec - execute command without creating new process	ksh(1)
exec - execute command without creating new process	sh-bourne(1)
exec - execute command without creating new process	sh-posix(1)
execl() , execle() , execlp() , execv() , execve() , execvp() - execute a file	exec(2)
executable and linking format object files (ELF)	elf(3E)
executable file, generate from object files and libraries	ld(1)
executable files in directory, list	ls(1)
executable, prepare for faster program start-up	fastbind(1)
execute a command on a remote host	rcmd(3N)
execute a command, measure time used to	time(1)
execute a file	exec(2)
execute a function, descending a directory tree	ftw(3C)
execute a regular expression against a string	regcmp(3X)
execute a simple command	command(1)
execute command on a remote host	on(1)
execute command with constructed large argument list(s)	xargs(1)
execute commands at a later time	at(1)
execute commands in background	at(1)
execute from a remote shell	remsh(1)
execute HALGOL programs	opx25(1M)
execute process with POSIX real-time priority	rtsched(1)
execute process with real-time priority	rtprio(1)
execute remote uucp or uux command requests on local system	uuxqt(1M)
execute, and compile, regular expressions	re_comp(3X)
executing program, send signal	kill(2)
execution of a thread, continue, resume, or suspend	pthread_resume_np(3T)
execution of commands, UNIX system to UNIX system	uux(1)
execution profile data, display call graph	gprof(1)
execution profile, prepare	monitor(3C)
execution server, remote	rexecd(1M)
execution server, RPC-based remote	rexcd(1M)
execution startup routines, C, Pascal, and FORTRAN	crt0(3)
execution time profile	profil(2)

Description	Entry Name(Section)
execution, commands, set environment for	env(1)
execution, suspend for a time interval	sleep(1)
execution, suspend for an interval	usleep(2)
execution, suspend for interval	sleep(3C)
existing file, truncate to zero for rewriting	creat(2)
exit - exit shell with exit status	csh(1)
exit - exit shell with exit status	ksh(1)
exit - exit shell with exit status	sh-bourne(1)
exit - exit shell with exit status	sh-posix(1)
exit status, do nothing and return zero or non-zero	true(1)
exit() , _exit() - terminate process	exit(2)
exit, register a function to be called at	atexit(2)
exp() , expf() - exponential functions	exp(3M)
exp2() - base-2 exponential function	exp2(3M)
expand , unexpand - expand tabs to spaces, and vice versa	expand(1)
expand or compress data	compress(1)
expand text line to specified length	newform(1)
expands the sendmail aliases, recursively	expand_alias(1)
expand_alias - recursively expands the sendmail aliases	expand_alias(1)
expansions, perform word	wordexp(3C)
expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
expf() , exp() - exponential functions	exp(3M)
explicit load of shared libraries	shl_load(3X)
explicit locking of streams within a multi-thread application	flockfile(3S)
expm1() - exponential functions	expm1(3M)
exponent and mantissa, split double-precision number into	frexp(3M)
exponent of a floating-point number, load	ldexp(3M)
exponent, radix-independent	logb(3M)
exponential function, base-2	exp2(3M)
exponential functions	exp(3M)
exponential functions	expm1(3M)
export an LVM volume group and its associated logical volumes	vgexport(1M)
export and unexport directories to PFS clients	pfs_exportfs(1M)
export directories to NFS clients	exportfs(1M)
export directories, to PFS clients	pfs_exports(5)
export - export variable names to environment of subsequent commands	ksh(1)
export - export variable names to environment of subsequent commands	sh-bourne(1)
export - export variable names to environment of subsequent commands	sh-posix(1)
export to NFS clients, directories to	exports(4)
exported file system information, access	exportent(3N)
exportent() - access exported file system information	exportent(3N)
exportfs - export and unexport directories to NFS clients	exportfs(1M)
exports - directories to export to NFS clients	exports(4)
expr - evaluate arguments as an expression	expr(1)
expression matching routines, regular	regcomp(3C)
expression or string, search a file for a	grep(1)
expression, regular, and pattern matching notation definitions	regexp(5)
expression, regular, compile and match routines	regexp(3X)
expression, regular, compile or execute against a string	regcmp(3X)
expressions, compile and execute regular expressions	re_comp(3X)
exrt_unregister() - library routines for registering servers	rpc_svc_reg(3N)
extend an LVM volume group by adding physical volumes	vgextend(1M)
extend file system size	extendfs(1M)
extend HFS file system size	extendfs_hfs(1M)
extend VxFS file system size	extendfs_vxfs(1M)
extended general terminal interface	termiox(7)
extendfs - extend file system size	extendfs(1M)
extendfs_hfs - extend HFS file system size	extendfs_hfs(1M)
extendfs_vxfs - extend VxFS file system size	extendfs_vxfs(1M)
extent attributes (VxFS), get	getext(1M)
extent attributes (VxFS), set	setext(1M)

Index

All Volumes

Description	Entry Name(Section)
extract error messages from C source into a file	mkstr(1)
extract mantissa and exponent from double-precision number	frexp(3M)
extract non-repeated lines from a file	uniq(1)
extract portions of path names	basename(1)
extract selected fields of each line in a file	cut(1)
extract strings from C programs to implement shared strings	xstr(1)
f1crash dump, uncompress a file in	cr_uncompress(3)
f1file: uncompress a file in a crash dump	cr_uncompress(3)
f1uncompress a file in a crash dump	cr_uncompress(3)
fabs() - absolute value functions	fabs(3M)
fabsf() - absolute value functions	fabs(3M)
facilities, interprocess communication, report status	ipcs(1)
facilities, software signal	sigvector(2)
factor, primes - factor a number, generate large primes	factor(1)
factor a number, generate large primes	factor(1)
fallback mechanism	service.switch(1M)
false - do nothing and return non-zero exit status	true(1)
false/true evaluate condition for	test(1)
family, Internet protocol	inet(7F)
fast disk storage, preallocate	prealloc(2)
fast find: list file names and statistics for VxFS file system	ff_vxfs(1M)
fastbind - prepare an incomplete executable for faster program start-up	fastbind(1)
faster program start-up	fastbind(1)
faster tape I/O	ftio(1)
fastmail - quick batch mail interface	fastmail(1)
fattach() - attach a STREAMS file descriptor	fattach(3C)
fault (interrupt) conditions, define for I/O device	io_on_interrupt(3I)
fault, generate an IOT	abort(3C)
fbackup - back up selected files or groups of files	fbackup(1M)
fc - edit and execute previous command	ksh(1)
fc - edit and execute previous command	sh-posix(1)
fchdir() - change working directory	chdir(2)
fchmod() - change file mode access permissions	chmod(2)
fchown() - change owner and group of a file	chown(2)
fclose() - flush buffer then close stream	fclose(3S)
fcmsutil - fibre channel diagnostic utility	fcmsutil(1M)
fcntl - file control options	fcntl(5)
fcntl() - open-file control	fcntl(2)
fcpacl() - copy access control list (ACL) to another file	cpacl(3C)
fcutil - fibre channel diagnostic utility	fcutil(1M)
fcvt(), ecvt() - convert floating-point number to string	ecvt(3C)
FDDI (PCI) interface status, show	fddipciadmin(1M)
fddi - Fiber Distributed Data Interface Tools	fddi(7)
FDDI interface status, show	fddistat(1M)
FDDI interface tools	fddi(7)
FDDI interface, initialize	fddiinit(1M)
FDDI interface, stop and reset	fddistop(1M)
FDDI network interfaces, initialize and connect all system	fddissetup(1M)
FDDI network, connect to the	fddiinit(1M)
FDDI ring, list characteristics of nodes on	fddinet(1M)
FDDI SNMP subagent daemon	fddisubagtd(1M)
fddiinit - connect to the FDDI network	fddiinit(1M)
fddinet - list characteristics of nodes on FDDI ring	fddiinit(1M)
fddipciadmin - show PCI FDDI interface status	fddipciadmin(1M)
fddissetup - initialize and connect all system FDDI network interfaces	fddissetup(1M)
fddistat - show FDDI interface status	fddistat(1M)
fddistop - stop and reset FDDI interface	fddistop(1M)
fddisubagtd - FDDI SNMP subagent daemon	fddisubagtd(1M)
fdetach - detach a name from a STREAMS file descriptor	fdetach(3C)
fdetach - detach a STREAMS-based file descriptor	fdetach(1M)
fdim() - positive difference function	fdim(3M)

Description	Entry Name(Section)
fdopen() - associate a stream with an open file descriptor	fopen(3S)
feclearexcept() - clear floating-point exceptions	feclearexcept(3M)
fegetenv() - get floating-point environment	fegetenv(3M)
fegetexceptflag() - get floating-point exception flags	fegetexceptflag(3M)
fegetflushtozero() - get floating-point underflow mode	fegetflushtozero(3M)
fegetround() - get floating-point rounding mode	fegetround(3M)
fegettrapenable() - get exception trap enable bits	fegettrapenable(3M)
feholdexcept() - save floating-point environment	feholdexcept(3M)
feof() - check for end-of-file error on stream	ferror(3S)
feraiseexcept() - raise floating-point exceptions	feraiseexcept(3M)
ferror() - check for I/O error on stream	ferror(3S)
fesetenv() - set floating-point environment	fesetenv(3M)
fesetexceptflag() - set floating-point exception flags	fesetexceptflag(3M)
fesetflushtozero() - set floating-point underflow mode	fesetflushtozero(3M)
fesetround() - set floating-point rounding mode	fesetround(3M)
fesettrapenable() - set exception trap enable bits	fesettrapenable(3M)
fetch() - access data under a key (old single-data-base version)	dbm(3X)
fetestexcept() - test floating-point exceptions	fetestexcept(3M)
feupdateenv() - update floating-point environment	feupdateenv(3M)
ff - fast find: list file names and statistics for VxFS file system	ff_vxfs(1M)
ff - list file names and statistics for file system (generic)	ff(1M)
ff - list file names and statistics for HFS file system	ff_hfs(1M)
fflush() - flush buffer without closing stream	fclose(3S)
ffs() - BSD find first set bit	memory(3C)
ff_vxfs - fast find: list file names and statistics for VxFS file system	ff_vxfs(1M)
fgetacl() - get access control list (ACL) information	getacl(2)
getc() - get character or word from a stream file	getc(3S)
fgetgrent() - get next entry in group() -file-formatted input stream	getgrent(3C)
fgetpos() - save file position indicator for a stream	fgetpos(3S)
fgetpos64() - file system API to support large files	fgetpos64(2)
fgetpwent() - get next entry in password-file-formatted input stream	getpwent(3C)
fgets() , gets() - get a string from a standard input stream	gets(3S)
fgetspwent() - get next entry in secure password-file-formatted input stream	getspwent(3X)
fgetc() - get wide character from a stream file	getwc(3C)
fgetwc_unlocked() - get wide character from a stream file	getwc(3C)
fgetws() - get a wide string from a stream file	fgetws(3C)
fgrep - search a file for a specific string (fast algorithm)	grep(1)
Fibre Distributed Data Interface, FDDI	fddi(7)
fibre channel diagnostic utility	fcmsutil(1M)
fibre channel diagnostic utility	fcutil(1M)
Fibre Channel Light Weight Protocol Echo Request packet, send	pong(1M)
Fibre Channel Light Weight Protocol, show network status	lwpstat(1M)
fields of each line in a file, cut out (extract) selected	cut(1)
FIFO (named pipe) special files, make	mkfifo(1)
FIFO files, create	mknod(1M)
FIFO scheduling policy	rtsched(2)
FIFO special file	glossary(9)
FIFO special file, make a	mkfifo(3C)
file access and modification times, set or update	utime(2)
file access and modification times, set	utimes(2)
file access mode	glossary(9)
file access permissions	glossary(9)
File Access software, Data Communications and Terminal Controller Device	ddfa(7)
file and data formats, audio	see audio(5)
file archiver, tape	tar(1)
file control options for open files	fcntl(5)
file copy, public UNIX system to UNIX system	uuto(1)
file copy, remote	rcp(1)
file copy, remote	rcp(1)
file creation (permissions) mask, set and get	umask(2)
file creation, set access permissions mode mask for	umask(1)

Index All Volumes

Description	Entry Name(Section)
file descriptor	glossary(9)
file descriptor, make, for ELF files	elf_begin(3E)
file descriptor, map stream pointer to	fileno(3S)
file descriptor: attach a STREAMS file descriptor	fattach(3C)
file descriptor: detach a name from a STREAMS-based file descriptor	fdetach(3C)
file descriptor: detach a STREAMS-based file descriptor	fdetach(1M)
file descriptor: duplicate an open file descriptor	dup(2)
file descriptor: duplicate an open file descriptor to a specific slot	dup2(2)
file descriptor: STREAMS device	isastream(3C)
file descriptor: STREAMS-based pipe	isastream(3C)
file - determine file type	file(1)
file distribution to remote hosts	rdist(1)
file editing activity, print current SCCS	sact(1)
file entry get file system description	getmntent(3X)
file entry, get file system descriptor (BSD 4.2 compatibility only)	getfsent(3X)
file for HPDPS gateway printers, configuration	pdgwcfg.conf(4)
file for localedef scripts, symbolic translation	charmap(4)
file for rexec() and ftp security	netrc(4)
file for the SNMP agent configuration	snmpd.conf(4)
file format and other information for auditing	audit(4)
file format for keysh softkeys	softkeys(4)
file format, <pwd.h> password file	passwd(4)
file format, ppp authentication	ppp.Auth(4)
file format, ppp dialer description	ppp.Dialers(4)
file format, ppp encryption keys	ppp.Keys(4)
file format, ppp neighboring systems description	ppp.Systems(4)
file format, ppp packet filter specification	ppp.Filter(4)
file format, ppp physical device description	ppp.Devices(4)
file format, translate host table to name server	hosts_to_named(1M)
file format: common archive files	ar(4)
file format: compiled terminfo file format	term(4)
file format: core image files	core(4)
file format: disk description file format	disktab(4)
file format: per-process accounting files	acct(4)
file format: Product Description File	pdf(4)
file format: protected password database	prpwd(4)
file format: Revision Control System (RCS) files	rcsfile(4)
file format: SCCS file format	scsfile(4)
file format: terminal control database	ttys(4)
file format: text file format specification	fspec(4)
file format: user accounting file entry	utmp(4)
file formats, introduction	intro(4)
file	glossary(9)
file group class	glossary(9)
file hierarchy	glossary(9)
file identification data for ELF files, retrieve	elf_getident(3E)
file link: soft (symbolic) link	symlink(4)
file locking, provide semaphores and record locking on files	lockf(2)
file mode, change access permissions	chmod(1)
file mode, change access permissions	chmod(2)
file name generation function	glob(3C)
file name	glossary(9)
file name of controlling terminal, generate	ctermid(3S)
file name portability	glossary(9)
file names and statistics for HFS file system, list	ff_hfs(1M)
file names and statistics for VxFS file system	ff_vxfs(1M)
file names for file system, list (generic)	ff(1M)
file names, long, convert a file system to allow	convertfs(1M)
file offset	glossary(9)
file on remote node, get file handle for	getfh(2)
file or anonymous memory region, initialize semaphore in mapped	msem_init(2)

Description	Entry Name(Section)
file or anonymous region, remove semaphore in mapped	msem_remove(2)
file or file structure, determine which processes are using	fuser(1M)
file other class	glossary(9)
file owner class	glossary(9)
file path, map device ID to	devnm(3)
file permission bits	glossary(9)
file perusal filter for CRT terminals	more(1)
file perusal filter for soft-copy terminals	pg(1)
file pointer	glossary(9)
file pointer: move read/write file pointer	lseek(2)
file position indicator for a stream, save or restore	fgetpos(3S)
file serial number	glossary(9)
file size in words, lines, and bytes or characters	wc(1)
file size limits and break value, get or set	ulimit(2)
file status flags	glossary(9)
file status, get	fstat(2)
file structures, statd directory and	sm(4)
file syntax, gated configuration	gated.conf(4)
file system (generic), construct	mkfs(1M)
file system (generic), construct new	newfs(1M)
file system (HFS), construct a new	newfs_hfs(1M)
file system (HFS), construct	mkfs_hfs(1M)
file system administration command	fsadm(1M)
file system administration command	fsadm_hfs(1M)
file system administration commands, configuration and binary files	fs_wrapper(5)
file system APIs to support large files	creat64(2)
file system APIs to support large files	fgetpos64(2)
file system clean at last system shutdown, test for	fsclean(1M)
file system consistency check and interactive repair	fsck(1M)
file system consistency check and interactive repair (generic)	fsck(1M)
file system control	fsctl(2)
file system debugger (generic)	fsdb(1M)
file system debugger (HFS)	fsdb_hfs(1M)
file system debugger, VxFS	fsdb_vxfs(1M)
file system description file entry	getmntent(3X)
file system disk blocks (generic), report number of free	df(1M)
file system disk blocks, report number of free CDFS, HFS, or NFS	df_hfs(1M)
file system for paging, enable	swapon(1M)
file system	glossary(9)
file system information, access exported	exportent(3N)
file system information, dump	dumpfs(1M)
file system mount statistics, local	rmtab(4)
file system mounting table, static	pfs_fstab(5)
file system name space: attach a STREAMS file descriptor	fattach(3C)
file system quota consistency checker, VxFS	quotacheck_vxfs(1M)
file system quota consistency checker, generic	quotacheck(1M)
file system quotas, turn on and off	quotaon(1M)
file system size, extend	extendfs(1M)
file system size, extend HFS	extendfs_hfs(1M)
file system size, extend VxFS	extendfs_vxfs(1M)
file system statistics, get	statfs(2)
file system statistics, get	statfsdev(3C)
file system statistics, get	statvfsdev(3C)
file system swapping	swapon(2)
file system type info, get	sysfs(2)
file system type, determine	fstyp(1M)
file system, check and repair VxFS	fsck_vxfs(1M)
file system, compare with Product Description File	pdfck(1M)
file system, construct VxFS	mkfs_vxfs(1M)
file system, copy HFS with compaction	dcopy(1M)
file system, damaged, patch up (generic)	fsdb(1M)

Index

All Volumes

Description

Entry Name(Section)

file system, damaged, patch up (HFS)	fsdb_hfs(1M)
file system, dump across network	vxdump(1M)
file system, generate pathnames from inode numbers for VxFS	ncheck_vxfs(1M)
file system, get mounted file system statistics	ustat(2)
file system, incremental dump (for backups)	dump(1M)
file system, list file names and statistics (generic)	ff(1M)
file system, list file names and statistics for (generic)	ff(1M)
file system, list file names and statistics for HFS	ff_hfs(1M)
file system, list file names and statistics for VxFS	ff_vxfs(1M)
file system, local dump	vxdump(1M)
file system, mount an LOFS	mount_lofs(1M)
file system, mount and unmount VxFS	mount_vxfs(1M)
file system, mount	mount(2)
file system, mount or unmount (generic)	mount(1M)
file system, mount	vfsmount(2)
file system, perform Network File System mount to remote	vhe_u_mnt(1M)
file system, portable, PFS	pfs(4)
file system, repair VxFS	fsck_vxfs(1M)
file system, report free disk blocks on VxFS	df_vxfs(1M)
file system, restore incrementally, local or across network	vxrestore(1M)
file system, restore incrementally, local or across network	restore(1M)
file system: backup or archive the file system	backup(1M)
file system: construct new VxFS file system	news_vxfs(1M)
file system: convert a file system to allow long file names	convertfs(1M)
file system: display disk usage and limits quota	quota(1)
file system: file system hierarchy	hier(5)
file system: file system volume format	fs(4)
file system: get file system descriptor file entry (BSD 4.2 compatibility only)	getfsent(3X)
file system: mounted file system table	mnttab(4)
file system: optimize an existing HFS file system	tunefs(1M)
file system: static information about file systems	fstab(4)
file system: summarize quotas	repquota(1M)
file system: tune an existing HFS file system	tunefs(1M)
file system: unmount a file system	umount(2)
file systems with label checking, copy	volcopy(1M)
file systems with label checking, copy	volcopy_hfs(1M)
file systems, automatically mount NFS	automount(1M)
file systems, CD-ROM, mount and unmount	pfs_mount(1M)
file systems, keep track of remotely mounted	mount(3N)
file systems, mount and unmount CDFS	mount_cdfs(1M)
file systems, mount and unmount HFS	mount_hfs(1M)
file systems, mount and unmount multiple	mountall(1M)
file systems, mount and unmount NFS	mount_nfs(1M)
file times update	glossary(9)
file transfer program	ftp(1)
file transfer program, trivial	tftp(1)
file transfer program, XMODEM-protocol	umodem(1)
file transfer protocol server	ftpd(1M)
file transfer protocol server, trivial	tftpd(1M)
file tree, walk, executing a function	ftw(3C)
file type for ELF files, determine	elf_kind(3E)
file used by DDFA software and Telnet port identification feature, dedicated ports	dp(4)
file's in-core state with its state on disk, synchronize a	fsync(2)
file, aliases file for sendmail	aliases(5)
file, change file mode access permissions	chmod(2)
file, change owner or group	chown(1)
file, change system configuration	ch_rc(1M)
file, configuration, for pluggable authentication module	pam.conf(4)
file, context-dependent	chmod(1)
file, context-dependent	chmod(2)
file, create a kernel system	create_sysfile(1M)

Description	Entry Name(Section)
file, create message catalog file for modification	findmsg(1)
file, create special (device)	mksf(1M)
file, create special and FIFO	mknod(1M)
file, determine accessibility	access(2)
file, directory, remove	rmdir(2)
file, driver: driver information for insf , mksf , lssf	devices(4)
file, dump window to and reload window from	getwin(3X)
file, for screen, input/output functions	scr_dump(3X)
file, generate a formatted message catalog	gencat(1)
file, get a user's effective access rights to a	getaccess(2)
file, get file status	stat(2)
file, get file status	statvfs(2)
file, group: get entry from group() file	getgrent(3C)
file, hidden	chmod(1)
file, hidden	chmod(2)
file, inetd configuration	inetd.conf(4)
file, inetd optional security	inetd.sec(4)
file, link additional name to an existing	link(2)
file, link existing file to a new file name	ln(1)
file, LVM physical volume group information	lvmpvg(4)
file, make a directory, special, or ordinary	mknod(2)
file, make a symbolic link	symlink(2)
file, open for reading or writing	open(2)
file, password: get entry from password file	getpwent(3C)
file, password: get entry from secure password file	getspent(3C)
file, password: get entry from secure password file	getspwent(3C)
file, print out mail in the incoming mailbox	prmail(1)
file, read data from	read(2)
file, remove a special (device)	rmsf(1M)
file, resolver configuration	resolver(4)
file, security for ftpd(1M)	ftpusers(4)
file, send the contents of a file through a socket	sendfile(2)
file, stream, get character or data word from a stream file	getc(3S)
file, stream: buffered binary input/output to a stream file	read(3S)
file, stream: convert file to stream; open or re-open a stream file	fopen(3S)
file, stream: get wide character from a stream file	getwc(3C)
file, stream: open or re-open a stream file; convert file to stream	fopen(3S)
file, stream: reposition or get pointer for I/O operations on a stream file	fseek(3S)
file, synchronize a mapped	msync(2)
file, used by DDFA software, configuration	pcf(4)
file, utmp() , of the current user, find the slot in the	ttyslot(3C)
file, validate an SCCS file	val(1)
file, Virtual Home Environment information	vhe_list(4)
file, write data	write(2)
file-name suffix conventions	suffix(5)
file: access utmpx() file	getutx(3C)
file: access wtmp() or utmp() file	getut(3C)
file: assign buffering to a stream file	setbuf(3S)
file: change file mode access permissions	chmod(1)
file: change owner and group	chown(2)
file: change the name of a file	rename(2)
file: close a file descriptor	close(2)
file: control a file descriptor for ELF files	elf_cntl(3E)
file: convert binary file to ASCII for transmission by mailer	uuencode(1)
file: copy a file into memory	copylist(3C)
file: copy access control list (ACL) to another file	cpacl(3C)
file: create a name for a temporary file	tmpnam(3S)
file: create a new file or rewrite an existing one	creat(2)
file: create a temporary file	tmpfile(3S)
file: create zero-length file	cat(1)
file: create zero-length file	cp(1)

Index All Volumes

Description	Entry Name(Section)
file: create zero-length file	null(7)
file: create zero-length file	touch(1)
file: decode a file encoded by uuencode	uuencode(1)
file: delete	unlink(2)
file: discard file (bit bucket)	null(7)
file: execute a file	exec(2)
file: get the base offset for an object file	elf_getbase(3E)
file: header file for future applications	portal(5)
file: issue (/etc/issue) identification file format	issue(4)
file: main memory and kernel memory image file	mem(7)
file: make a unique (usually temporary) file name	mktemp(3C)
file: null file (bit bucket)	null(7)
file: object file access library	elf(3E)
file: open-file control routines	fcntl(2)
file: read from file, stream, or character string with formatted input conversion	scanf(3S)
file: receive next message from a STREAMS file	getmsg(2)
file: remove a file	remove(3C)
file: remove nroff/troff, tbl, and neqn constructs from	deroff(1)
file: return the size of an object file type for elf32 or elf64 files	elf_size(3E)
file: rewrite an existing file	creat(2)
file: search for named file in named directories	pathfind(3G)
file: special (device) files	(see special files)
file: truncate a file to a specified length	truncate(2)
file: truncate an existing file to zero for rewriting	creat(2)
file: user accounting information file	utmpx(4)
filename patterns, match	fnmatch(3C)
filename: detach a STREAMS-based file descriptor	fdetach(1M)
fileno() - map stream pointer to file descriptor	fileno(3S)
files or records, move magnetic tape forward or backward by	mt(1)
files, accounting, search and print	acctcom(1M)
files, accounting: convert process accounting files to ASCII text format	acctprc(1M)
files, accounting: merge or add total accounting files	acctmerg(1M)
files, accounting: print session record file created by acctcon1	acctsh(1M)
files, accounting: summarize process accounting files created by acctprc1	acctprc(1M)
files, administer and create SCCS files	admin(1)
files, audit, set or get	audctl(2)
files, C header, generate	rpcgen(1)
files, change name of a file	mv(1)
files, configuration and binary, file system administration	fs_wrapper(5)
files, Cyclical Redundancy Check on a file	sum(1)
files, find the printable strings in an object or other binary file	strings(1)
files, format and print	pr(1)
files, install special (device)	insf(1M)
files, move directory subtree and files to another directory	mv(1)
files, move file to new location	mv(1)
files, move multiple files to another directory	mv(1)
files, object code: find ordering relation for files in an object code library	lorder(1)
files, object code: install in binary directories	cpset(1M)
files, object code: optimum sequence for object code files in a library, find	lorder(1)
files, overwrite file with an existing file	mv(1)
files, print checksum and block count of a file	sum(1)
files, remove outdated STREAMS error log files	strclean(1M)
files, rename directory	mv(1)
files, rename file	mv(1)
files, search a file for a string or expression	grep(1)
files, send copy of standard output to specified file	tee(1)
files, send to system log	logger(1)
files, text: format text file for CRT or line-printer output	nroff(1)
files, user configuration files for pluggable authentication modules	pam_user.conf(4)
files: accounting summary files, create periodic	acctsh(1M)
files: add line number in front of each line in a file	nl(1)

Description	Entry Name(Section)
files: back up selected files or groups of files	fbackup(1M)
files: big file scanner	bfs(1)
files: break a single file into multiple files	split(1)
files: change or reformat a text file	newform(1)
files: check nroff/troff files	checknr(1)
files: compare two files and mark differences	diffmk(1)
files: compare two files and show differences side-by-side	sdiff(1)
files: compare two files	cmp(1)
files: compare two versions of an SCCS file	sccsdiff(1)
files: compress data in a file	compress(1)
files: compress files in a directory	compress(1)
files: context split	csplit(1)
files: convert DOS file to HP-UX ASCII file format	dos2ux(1)
files: convert file keyboard/display data order	forder(1)
files: convert HP-UX ASCII file to DOS file format	dos2ux(1)
files: copy directory subtree and files to another directory	cp(1)
files: copy file archives in and out	cpio(1)
files: copy file to a new or existing file	cp(1)
files: copy multiple files to a directory	cp(1)
files: copy to or from DOS files	doscp(1)
files: copy to or from remote system	rcp(1)
files: copy to or from remote system	rcp(1)
files: count words, lines, and bytes or characters in a file	wc(1)
files: create a tags file	ctags(1)
files: create the cat files for the manual	catman(1M)
files: cut out (extract) selected fields of each line in a file	cut(1)
files: determine file type	file(1)
files: differential file comparator	diff(1)
files: display file on CRT terminals	more(1)
files: display file on soft-copy terminals	pg(1)
files: dump file in octal or hexadecimal format	od(1)
files: eliminate adjacent repeated lines in a file	uniq(1)
files: encrypt/decrypt files	crypt(1)
files: error message file, extract error messages from C source into	mkstr(1)
files: expand compressed file	compress(1)
files: expand compressed files in a directory	compress(1)
files: find (search for) files	find(1)
files: find differences among three files	diff3(1)
files: find differences between two files	diff(1)
files: format tracing and logging binary files	netfmt(1M)
files: get a version of an SCCS file	get(1)
files: get first few lines in a file	head(1)
files: get lines from last part of a file	tail(1)
files: get SCCS identification information from files	what(1)
files: get status	fstat(2)
files: job file, prototype, for at	proto(4)
files: list access control lists (ACLs) of files	lsacl(1)
files: list access rights to file(s)	getaccess(1)
files: make a delta (change) to an SCCS file	delta(1)
files: make unprintable and non-ASCII characters in a file visible or invisible	vis(1)
files: merge corresponding lines of several files or subsequent lines of one file	paste(1)
files: name for a temporary file, make a	mktemp(1)
files: network tracing and logging configuration file	netlgen.conf(4)
files: overwrite file with an existing file	cp(1)
files: password file, edit using vi editor	vipw(1M)
files: print and summarize an SCCS file	prs(1)
files: print checksum and block count of a file	sum(1)
files: print first few lines in a file	head(1)
files: print section sizes and allocation space of object files	size(1)
files: prototype job file for at	proto(4)
files: queue description file for at , batch , and crontab	queuedefs(4)

Index

All Volumes

Description	Entry Name(Section)
files: read portable archive	pax(1)
files: reduce multiple adjacent blank lines to single blank line	ssp(1)
files: remove a delta from an SCCS file	rmidel(1)
files: remove all blank lines from file	rmnl(1)
files: remove DOS files or directories	dosrm(1)
files: remove file that is not listed in any directory	clri(1M)
files: remove	rm(1)
files: report adjacent repeated lines in a file	uniq(1)
files: reverse the left-to-right text character sequence in each line of a file	rev(1)
files: schedule uucp transport files	uusched(1M)
files: select/reject lines common to two sorted files	comm(1)
files: selectively recover files from backup media	frecover(1M)
files: sort and/or merge files	sort(1)
files: split a file into multiple <i>n</i> -line pieces	split(1)
files: split file into multiple files	csplit(1)
files: strip symbol and line number information from an object file	strip(1)
files: temporary file, make a name for a	mktemp(1)
files: three-way differential file comparator	diff3(1)
files: three-way file merge	merge(1)
files: transfer uucp-system files in or out	uucico(1M)
files: undo a previous get of an SCCS file	unget(1)
files: user crontab file operations	crontab(1)
files: verify internal revision numbers of HP-UX files	revck(1M)
files: write portable archive	pax(1)
filter	glossary(9)
filter reverse line-feeds and backspaces from text	col(1)
filter() - disable use of certain terminal capabilities	filter(3X)
filter, line numbering	nl(1)
find adjacent repeated lines in a file	uniq(1)
find differences among three files	diff3(1)
find differences between two files	diff(1)
find - find (search for) files	find(1)
find hyphenated words	hyphen(1)
find location of source, binary, and/or manual files for program	whereis(1)
find manual information by keywords; print out a manual entry	man(1)
find name of a terminal	ttyname(3C)
find ordering relation for files in an object code library	lorder(1)
find program files that execute under given command name	which(1)
find spelling errors	spell(1)
find strings for inclusion in message catalogs	findstr(1)
find the printable strings in an object or other binary file	strings(1)
find the slot in the utmp() file of the current user	ttyslot(3C)
findmsg - create message catalog file for modification	findmsg(1)
findstr(1) output, use to insert calls to catgets(3C)	insertmsg(1)
findstr - find strings for inclusion in message catalogs	findstr(1)
finger , change information in password file	chfn(1)
finger - user information lookup program	finger(1)
fingerd - remote user information server	fingerd(1M)
finish using an ELF object file	elf_end(3E)
finish, wait for background processes to	wait(1)
finite-width output device, fold long lines for	fold(1)
finiteness macro, floating-point	isfinite(3M)
firmware (processor-dependent code)	pdc(1M)
first locations beyond allocated program regions	end(3C)
firstkey() - get first key in database (old single-data-base version)	dbm(3X)
firstof2() , FIRSTOF2() - test for valid first byte in 16-bit character	nl_tools_16(3X)
fix damaged file system (generic)	fsdb(1M)
fix damaged HFS file system	fsdb_hfs(1M)
fixman - fix manual pages for faster viewing with man(1)	fixman(1M)
flag for calling process, get audit process	getaudproc(2)
flags, set, query configuration and loadable flags for a module	kmsystem(1M)

Description	Entry Name(Section)
flash the screen	flash(3X)
flash() - flash the screen	flash(3X)
flexible or "floppy" disk device driver	floppy(7)
floating-point classification macros	fpclassify(3M)
floating-point classification macros	isfinite(3M)
floating-point classification macros	isinf(3M)
floating-point classification macros	isnan(3M)
floating-point classification macros	isnormal(3M)
floating-point comparison macro (<)	isless(3M)
floating-point comparison macro (<=)	islessequal(3M)
floating-point comparison macro (<=)	islessgreater(3M)
floating-point comparison macro (>)	isgreater(3M)
floating-point comparison macro (>=)	isgreaterequal(3M)
floating-point comparison macro (unordered)	isunordered(3M)
floating-point environment macros and functions	fenv(5)
floating-point environment: getting	fegetenv(3M)
floating-point environment: saving	fehldexcept(3M)
floating-point environment: setting	fesetenv(3M)
floating-point environment: updating	feupdateenv(3M)
floating-point exception flags: getting	fegetexceptflag(3M)
floating-point exception flags: setting	fesetexceptflag(3M)
floating-point exceptions: clearing	feclearexcept(3M)
floating-point exceptions: getting trap enable bits	fegettrapenable(3M)
floating-point exceptions: raising	feraiseexcept(3M)
floating-point exceptions: setting trap enable bits	fesettrapenable(3M)
floating-point exceptions: testing	fetestexcept(3M)
floating-point number to string, convert long double	ldcv(3C)
floating-point number, load exponent of a	ldexp(3M)
floating-point number: decompose	modf(3M)
floating-point rounding mode: getting	fegetround(3M)
floating-point rounding mode: setting	fesetround(3M)
floating-point sign-determination macro	signbit(3M)
floating-point underflow mode: getting	fegetflushtozero(3M)
floating-point underflow mode: setting	fesetflushtozero(3M)
floating-point: convert floating-point number to string or string array element	ecvt(3C)
floating-point: extract mantissa and exponent from double-precision number	frexp(3M)
floating-point: load exponent of a radix-independent floating-point number	scalb(3M)
floating-point: load exponent of a radix-independent floating-point number	scalbn(3M)
flockfile() , funlockfile() - explicit locking of streams within a multi-thread application	flockfile(3S)
floor function	floor(3M)
floor() - floor function	floor(3M)
floppy - flexible or "floppy" disk device driver	floppy(7)
flush buffer with or without closing stream	fclose(3S)
flush buffers to disk	sync(2)
flush unwritten system buffers to disk	sync(1M)
flush, enable or disable on interrupt	intrflush(3X)
flushing queue, enable/disable	noqiflush(3X)
flushinp() - discard input	flushinp(3X)
fmax() - maximum value function	fmax(3M)
fmin() - minimum value function	fmin(3M)
fmod() - remainder functions	fmod(3M)
fmodf() - remainder functions	fmod(3M)
fmt - format text	fmt(1)
fmtmsg() - displays formatted message on standard error and console	fmtmsg(3C)
fnmatch() - match filename patterns	fnmatch(3C)
fold - fold long lines for finite-width output device	fold(1)
fontdl - download fonts to printer	lpfilter(1)
fopen() - open a named file and associate with a stream	fopen(3S)
fopen64() - file system API to support large files	fgetpos64(2)
for - execute a do list	ksh(1)
for - execute a do list	sh-posix(1)

Index

All Volumes

Description	Entry Name(Section)
force process to relinquish processor	rtsched(2)
force propagation of a Network Information Service database	yppush(1M)
force target process to run serially with other processes	serialize(2)
foreach - initiate repetitive loop	csh(1)
foreground process group	glossary(9)
foreground process group ID	glossary(9)
foreground process group ID, get	tcgetpgrp(3C)
foreground process group ID, set	tcsetpgrp(3C)
fork	glossary(9)
fork handler	pthread_atfork(3T)
fork() - create a new process	fork(2)
format and print arguments	printf(1)
format and print files	pr(1)
format and semantics, localedef -command input script	localedef(4)
format date and time, convert user	getdate(3C)
format - format an HP SCSI disk array LUN	format(1M)
format of a CDFS cnode	cnode(4)
format of an inode	inode(4)
format of directory streams and directory entries	dirent(5)
format of HP-UX directory streams	ndir(5)
format of VxFS file system volume	fs_vxfs(4)
format of VxFS inode	inode_vxfs(4)
format specification in text files	fspec(4)
format text	fmt(1)
format tracing and logging binary files	netfmt(1M)
format, ioconfig entry	ioconfig(4)
format, production specification file	swpackage(4)
format, PSF	swpackage(4)
format, tar tape archive	tar(4)
format, translate host table file to name server	hosts_to_named(1M)
format: common archive file	ar(4)
format: convert DOS file to HP-UX ASCII file format	dos2ux(1)
format: convert HP-UX ASCII file to DOS file format	dos2ux(1)
format: core image file	core(4)
format: cpio archive	cpio(4)
format: directories	dir(4)
format: file system volume	fs(4)
format: format mathematical text for nroff	neqn(1)
format: format text file for CRT or line-printer output	nroff(1)
format: per-process accounting files	acct(4)
format: privileged values	privgrp(4)
format: Product Description File format	pdf(4)
format: symbol table structure	nlist(4)
format: user accounting files btmpt() , utmp() , and wtmp()	utmp(4)
format: uuencode(1) -encoded file	uuencode(4)
formatted input conversion to a varargs argument	vscanf(3S)
formatted input conversion, read from stream file or character string	scanf(3S)
formatted input, convert from a window	vwscanw(3X)
formatted input, convert from a window	vw_scanw(3X)
formatted input, convert, from a window	mvscanw(3X)
formatted message catalog file, generate a	gencat(1)
formatted message, displays on standard error and console	fmtmsg(3C)
formatted output of a varargs argument list, print	vprintf(3S)
formatted output, print in a window	vwprintw(3X)
formatted output, print in a window	vw_printw(3X)
formatted output, print in window	mvprintw(3X)
formatted output, print to standard output, file, or string	printf(3S)
formatted read and conversion from stream file or character string	scanf(3S)
formatters, text	adjust(1)
formatters: check or print documents formatted with the mm macros	mm(1)
formatters: format text file for CRT or line-printer output	nroff(1)

Description	Entry Name(Section)
formatting conventions, numeric, of current locale, query	localeconv(3C)
formatting documents, MM macro package for	mm(5)
formatting manpages, macro package for	man(5)
formatting routines, define label for	setlabel(3)
FORTRAN execution startup routines	crt0(3)
fpathconf() - get configurable path name variables	pathconf(2)
fpclassify() - floating-point classification macro	fpclassify(3M)
fprintf() - print formatted output to a file	printf(3S)
fputc() , putc() - put character on a stream	putc(3S)
fputs() - write null-terminated string to a named stream file	puts(3S)
fputwc() , putwc() - put wide character on a stream	putwc(3C)
fputws() - write null-terminated wide string to a named stream file	putws(3C)
frame-buffer raster display device access	framebuf(7)
framebuf - information for raster frame-buffer devices	framebuf(7)
fread() , fwrite() - buffered binary input/output to a stream file	fread(3S)
frecover - selectively recover files from backup media	frecover(1M)
free a per-process timer	rmtimer(3C)
free disk blocks on a VxFS file system	df_vxfs(1M)
free disk blocks, report number of (Berkeley version)	bdf(1M)
free DOS disk clusters, report number of	dosdf(1)
free memory associated with word expansions	wordexp(3C)
free memory for library structure (X/OPEN TLI-XTI)	t_free(3)
free space percentage, dump file system	dumpfs(1M)
free() - release allocated block of main memory	malloc(3C)
freedisk - recover disk space	freedisk(1M)
freenetconfig() - get network configuration data base entry	getnetconfig(3N)
freopen() - substitute a named file in place of an already open stream	fopen(3S)
freopen64() - file system API to support large files	fgetpos64(2)
frexp() - extract mantissa and exponent from double-precision number	frexp(3M)
from - who is my mail from?	from(1)
from, who is mail from	mailfrom(1)
frt0.o , mfrt0.o - FORTRAN execution startup routines	crt0(3)
fs - format of file system volume	fs(4)
fsadm - file system administration command	fsadm(1M)
fsadm_hfs - HFS file system administration command	fsadm_hfs(1M)
fsadm_vxfs - resize or reorganize a VxFS file system	fsadm_vxfs(1M)
fscanf() - formatted read from named input stream file	scanf(3S)
fscat - cat a VxFS file system	fscat_vxfs(1M)
fscat_vxfs - cat a VxFS file system	fscat_vxfs(1M)
fsck(1M) , make a lost+found directory for	mklost+found(1M)
fsck - check and repair VxFS file system	fsck_vxfs(1M)
fsck - file system consistency check and interactive repair	fsck(1M)
fsck - file system consistency check and interactive repair (generic)	fsck(1M)
fsck_vxfs - check and repair VxFS file system	fsck_vxfs(1M)
fsckclean - determine shutdown status of specified file system	fsckclean(1M)
fsctl() - file system control	fsctl(2)
fsdb - file system debugger (generic)	fsdb(1M)
fsdb - HFS file system debugger	fsdb_hfs(1M)
fsdb - VxFS file system debugger	fsdb_vxfs(1M)
fsdb_vxfs - VxFS file system debugger	fsdb_vxfs(1M)
fseek() , rewind() , ftell() - reposition a file pointer in a stream	fseek(3S)
fseek() - set position of next I/O operation on stream file	fseek(3S)
fseeko() - set position of next I/O operation on stream file, non-POSIX API	fseek(3S)
fseeko64() - file system API to support large files	fgetpos64(2)
fseek_unlocked() - set position of next I/O operation on stream file, no locking of stream for multi-thread applications	fseek(3S)
fsetacl() - set access control list (ACL) information	setacl(2)
fsetaclentry() - add, modify, or delete access control list entry	setaclentry(3C)
fsetpos() - restore file position indicator for a stream	fgetpos(3S)
fsetpos64() - file system API to support large files	fgetpos64(2)
fsirand : install random inode generation numbers	fsirand(1M)

Description	Entry Name(Section)
fspec - format specification in text files	fspec(4)
fstab - static information about the file systems	fstab(4)
fstat/stat/lstat system call, data returned by	stat(5)
fstat() - get file status	fstat(2)
fstat64() - file system API to support large files	creat64(2)
fstatfs(), statfs() - get file system statistics	statfs(2)
statfsdev(), statfsdev() - get file system statistics	statfsdev(3C)
statvfs(), (statvfs()) - get open file status	statvfs(2)
statvfsdev(), statvfsdev() - get file system statistics	statvfsdev(3C)
statvfsdev64() - file system API to support large files	fgetpos64(2)
fstyp - determine file system type	fstyp(1M)
fsync(), fdatasync() - synchronize a file's in-core state with its state on disk	fsync(2)
fs_vxfs - format of VxFS file system volume	fs_vxfs(4)
fs_wrapper - configuration and binary files used by file system administration commands	fs_wrapper(5)
ftell() - get offset from beginning-of-file of current byte in stream file	fseek(3S)
ftello() - get offset from beginning-of-file of current byte in stream file, non-POSIX API	fseek(3S)
ftello64() - file system API to support large files	fgetpos64(2)
ftell_unlocked() - get offset from beginning-of-file of current byte in stream file, no locking of stream for multi-thread applications	fseek(3S)
ftime() - get date and time more precisely (Version 7 compatibility only)	ftime(2)
ftio - faster tape I/O	ftio(1)
ftok() - create interprocess communication identifier	ftok(3C)
ftp , login information for	netrc(4)
ftp - file transfer program	ftp(1)
ftpd , security file for	ftpusers(4)
ftpd - file transfer protocol server	ftpd(1M)
ftpusers - security file for <i>ftpd</i> (1M)	ftpusers(4)
ftruncate() - truncate a file to a specified length	truncate(2)
ftw() - walk a file tree executing a function	ftw(3C)
ftw64() - file sysmmmaptem API to support large files	fgetpos64(2)
full name: in elm aliases	newalias(1)
function key codes, get from a terminal	getn_wstr(3X)
function keys, enable/disable abbreviation of	keypad(3X)
function number	glossary(9)
function to be called at program termination, register a	atexit(2)
function, enhanced pad management	subpad(3X)
function, execute descending a directory tree	ftw(3C)
function, relative window creation	derwin(3X)
function, window refresh control	touchwin(3X)
functions and constants, math	math(5)
functions for screen file input/output	scr_dump(3X)
functions of HP 2640- and HP 2621-series terminals, handle special	hp(1)
functions, access NIS+ database	nis_db(3N)
functions, allow signals to interrupt	siginterrupt(2)
functions, floating-point environment macros and	fenv(5)
functions, for input mode control	cbreak(3X)
functions, for line update status	redrawwin(3X)
functions, for pad management	newpad(3X)
functions, misc NIS+	nis_servers(3N)
functions, NIS+ group manipulation functions	nis_groups(3N)
functions, NIS+ namespace	nis_names(3N)
functions, NIS+ tables	nis_tables(3N)
functions, query, for terminal insert and delay capability	has_ic(3X)
functions, screen initialisation functions	initscr(3X)
functions, soft label	slk_attoff(3X)
functions, window creation functions	newwin(3X)
functions, window cursor location	move(3X)
functions, window refresh control	is_linetouched(3X)
funlockfile(), flockfile() - explicit locking of streams within a multi-thread application	flockfile(3S)
fuser - list processes using a file or file structure	fuser(1M)
fwrite(), fread() - buffered binary input/output to a stream file	fread(3S)

Description	Entry Name(Section)
fwtmp - manipulate connect accounting records	fwtmp(1M)
gamma function, log	lgamma(3M)
gamma() , lgamma() , lgamma_r() , signgam() - log gamma function	lgamma(3M)
gated configuration file syntax	gated.conf(4)
gated configuration guide	gated.conf(4)
gated - gateway routing daemon	gated(1M)
gated's operational user interface	gdc(1M)
gated-config - gated configuration file syntax	see gated.conf(4)
gated.conf - GateD configuration guide	gated.conf(4)
gateway routing daemon	gated(1M)
gateways, monitor OSPF	ospf_monitor(1M)
gateways, query RIP	ripquery(1M)
gcrt0.o , gfrt0.o - C and Pascal execution startup routines	crt0(3)
gcvt() - convert floating-point number to string array element	ecvt(3C)
gcvt() - convert floating-point number to string array element	ecvt(3C)
gdc - operational user interface for gated	gdc(1M)
gencat - generate a formatted message catalog file	gencat(1)
general terminal interface	termio(7)
general terminal interface, extended	termiox(7)
generate a DES encryption key	makekey(1)
generate a formatted message catalog file	gencat(1)
generate a locale environment	localedef(1M)
generate an IOT fault	abort(3C)
generate C header files	rpcgen(1)
generate file name of controlling terminal	ctermid(3S)
generate file names	glob(3C)
generate hashing encryption	crypt(3C)
generate hashing encryption on large strings	bigcrypt(3C)
generate iconv translation tables	genxlt(1)
generate large primes, factor a number	factor(1)
generate path names from i-numbers	ncheck(1M)
generate pathnames from inode numbers for VxFS file system	ncheck_vxfs(1M)
generate permuted index	ptx(1)
generate printable representation of a character	unctrl(3X)
generate printable representation of a wide character	wunctrl(3X)
generate RPC protocols, C header files	rpcgen(1)
generate uniformly distributed pseudo-random numbers	drand48(3C)
generator, simple random-number	rand(3C)
generic file system debugger	fsdb(1M)
generic file system quota consistency checker	quotacheck(1M)
generic file system, construct	mkfs(1M)
generic file system, mount or unmount	mount(1M)
generic I/O device control commands	ioctl(5)
generic new file system, construct	newfs(1M)
genxlt - generate iconv translation tables	genxlt(1)
geocustoms - configure system language on multi-language systems	geocustoms(1M)
get a multi-byte character length limited string from the terminal	getnstr(3X)
get a multi-byte character string from the terminal	getstr(3X)
get a single-byte character from the terminal	getch(3X)
get a user's effective access rights to a file	getaccess(2)
get a wide character from a terminal	get_wch(3X)
get a wide character string and rendition from a cchar_t	getcchar(3X)
get access control list (ACL) information	getacl(2)
get additional cursor and window coordinates	getbegyx(3X)
get address of connected peer	getpeername(2)
get address of symbol in shared object	dlsym(3C)
get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
get an identifier for the current host	gethostid(2)
get and set concurrency level of unbound threads	pthread_getconcurrency(3T)
get and set current user context	getcontext(2)
get and set the priocellng attribute	pthread_mutexattr_getprotocol(3T)

Index

All Volumes

Description

Entry Name(Section)

get and set the prioceiling of a mutex	pthread_mutex_getprioceiling(3T)
get and set the protocol attribute	pthread_mutexattr_getprotocol(3T)
get and set the scheduling policy and associated parameters	pthread_getschedparam(3T)
get and set the spin attribute	pthread_mutexattr_getspin_np(3T)
get and set the thread-specific data associated with a key	pthread_getspecific(3T)
get and set the yield frequency attribute	pthread_mutexattr_getspin_np(3T)
get and/or set signal stack context	sigstack(2)
get attributes for pthread	pthread_attr_getdetachstate(3T)
get audit ID (aid()) for current process	getaudit(2)
get audit process flag for calling process	getaudproc(2)
get configurable path name variables	pathconf(2)
get configurable system variables	sysconf(2)
get current value of system-wide clock	getclock(3C)
get cursor and window coordinates	getyx(3X)
get disk description by its name	getdiskbyname(3C)
get entries from a directory in a file-system-independent format	getdirent(2)
get events and system calls currently being audited	getevent(2)
get extent attributes (VxFS)	getext(1M)
get file handle for file on remote node	getfh(2)
get file handle for file on remote node, get	getfh(2)
get file status	fstat(2)
get file status	statvfs(2)
get file system description file	getmntent(3X)
get file system statistics	statfs(2)
get file system statistics	statfsdev(3C)
get file system statistics	statvfsdev(3C)
get file system type info	sysfs(2)
get first few lines in a file	head(1)
get foreground process group ID	tcgetpgrp(3C)
get - get a version of an SCCS file	get(1)
get information about computer system	uname(2)
get information about resource utilization	getrusage(2)
get information about shared library	shl_load(3X)
get information for a dynamically loaded kernel module	modstat(2)
get information for a global kernel symbol	getksym(2)
get legal user shells	getusershell(3C)
get lines from last part of a file	tail(1)
get locale-specific (NLS) information	locale(1)
get login name	logname(1)
get mounted file system statistics	ustat(2)
get name of current NIS domain	getdomainname(2)
get name of key	keyname(3X)
get name of the user's terminal or pseudo-terminal	tty(1)
get name of user logged in on this terminal	getlogin(3C)
get network entry	getnetent(3N)
get network group entry	getnetgrent(3C)
get network host entry	gethostent(3N)
get network status	netstat(1)
get of an SCCS file, undo a previous	unget(1)
get or set audit files	audctl(2)
get or set background character and rendition using a complex character	bkgrnd(3X)
get or set background character and rendition using a single-byte character	bkgd(3X)
get or set the process-shared attribute	pthread_mutexattr_getpshared(3T)
get or set the process-shared attribute	pthread_rwlockattr_getpshared(3T)
get or set the thread process-shared attribute	pthread_condattr_getpshared(3T)
get or set the type attribute	pthread_mutexattr_getpshared(3T)
get or set tty baud rate	cfspeed(3C)
get PAM error message string	pam_strerror(3)
get pathname of current working directory	getwd(3C)
get process priority	getpriority(2)
get process, process group, or parent process ID	getpid(2)

Description	Entry Name(Section)
get protocol entry	getprotoent(3N)
get RPC entry	getrpcent(3C)
get RPC port number	getrpcport(3N)
get section information for ELF files	elf_getscn(3E)
get service entry	getservent(3N)
get signal alternate stack context	sigaltstack(2)
get socket address	getsockname(2)
get special attributes for group	getprivgrp(2)
get status information and attributes associated with a message queue	mq_getattr(2)
get supported terminal video attributes	termattrs(3X)
get system clock date and time	gettimeofday(2)
get system resource consumption limit	getrlimit(2)
get terminal baud rate	baudrate(3X)
get terminal name	termname(3X)
get Terminal Session Manager state information	tsm.info(1)
get the base offset for an object file	elf_getbase(3E)
get the current page size	getpagesize(2)
get the locale of a program	setlocale(3C)
get the name of a slave pty	ptsname(3C)
get tty device operating parameters	tcattribute(3C)
get value of a per-process timer	gettimer(3C)
get verbose description of current terminal	longname(3X)
get X.25 line	getx25(1M)
get, character or data word from a stream file	getc(3S)
get, file status	stat(2)
get, NLS program message	catgets(3C)
get, or put bootptab entry -	getbootpent(3X)
get, pointer for I/O operations on a stream file, get or reposition	fseek(3S)
get, symbolic link status	lstat(2)
get: data pointer for binary search tree	tsearch(3C)
get: date and time more precisely (Version 7 compatibility only)	ftime(2)
get: entries from name list	nlist(3C)
get: entry from group() file	getgrent(3C)
get: file size limits and break value, get or set	ulimit(2)
get: file system descriptor file entry (BSD 4.2 compatibility only)	getfsent(3X)
get: message queue	msgget(2)
get: name of current host	gethostname(2)
get: option letter from argument vector	getopt(3C)
get: path-name of current working directory	getcwd(3C)
get: process and child process times	times(2)
get: real or effective user or group ID	getuid(2)
get: set of semaphores	semget(2)
get: shared memory segment	shmget(2)
get: time	time(2)
get: value of process interval timer	getitimer(2)
get: wide character from a stream file	getwc(3C)
getaccess - list access rights to file(s)	getaccess(1)
getaccess() - get a user's effective access rights to a file	getaccess(2)
getacl() , fgetacl() - get access control list (ACL) information	getacl(2)
getaudit(2) - HP-UX Auditing System described	audit(5)
getauid() - get audit ID (auid()) for current process	getauid(2)
getaudproc() - get audit process flag for calling process	getaudproc(2)
getbegyx() - get additional cursor and window coordinates	getbegyx(3X)
getc() - get character or word from a stream file	getc(3S)
getcchar() - get a wide character string and rendition from a cchar_t	getcchar(3X)
getch() - get a single-byte character from the terminal	getch(3X)
getchar() - get character or word from standard input file	getc(3S)
getchar_unlocked() - get character or word from standard input file	getc(3S)
getclock() - get current value of system-wide clock	getclock(3C)
getconf - get POSIX configuration values	getconf(1)
getcontext() - get and set current user context	getcontext(2)

Description

Entry Name(Section)

getcwd() - get path-name of current working directory	getcwd(3C)
getc_unlock() - get character or word from a stream file	getc(3S)
getdate() - convert user format date and time	getdate(3C)
getdate_r() - convert user format date and time	getdate(3C)
getdirenties() - get entries from a directory in a file-system-independent format	getdirenties(2)
getdiskbyname() - get disk description by its name	getdiskbyname(3C)
getdomainname() - get name of current NIS domain	getdomainname(2)
getdvagent() - return pointer for device assignment database entry	getdvagent(3)
getdvagname() - return success or failure information	getdvagent(3)
getegid() - get effective group ID	getuid(2)
getenv() - return value for environment name	getenv(3C)
geteuid() - get effective user ID	getuid(2)
getevent(2) - HP-UX Auditing System described	audit(5)
getevent() - get events and system calls currently being audited	getevent(2)
getexportent() - access exported file system information	exportent(3N)
getexportopt() - access exported file system information	exportent(3N)
getext - get extent attributes (VxFS)	getext(1M)
getfh() - get file handle for file on remote node	getfh(2)
getfsent() - get next line in file system descriptor file	getfsent(3X)
getfsfile() - search descriptor file for ordinary file entry	getfsent(3X)
getfsspec() - search descriptor file for special (device) file entry	getfsent(3X)
getfstype() - search descriptor file for specified file type entry	getfsent(3X)
getgid() - get real group ID	getuid(2)
getgrent() - get next entry in group() file	getgrent(3C)
getgrgid() - get entry from group() file that matches gid()	getgrent(3C)
getgrnam() - get entry from group() file that matches group name name()	getgrent(3C)
getgroups() - get group access list	getgroups(2)
gethostbyaddr() - get network host entry	gethostent(3N)
gethostbyaddr_r() - get network host entry (thread-safe)	gethostent(3N)
gethostbyname() - get network host entry	gethostent(3N)
gethostbyname_r() - get network host entry (thread-safe)	gethostent(3N)
gethostent() - get network host entry	gethostent(3N)
gethostent_r() - get network host entry (thread-safe)	gethostent(3N)
gethostid() - get an identifier for the current host	gethostid(2)
gethostname() - get name of current host	gethostname(2)
getitimer() - get value of process interval timer	getitimer(2)
getksym() - get information for a global kernel symbol	getksym(2)
getlocale() - get the locale of a program	setlocale(3C)
getlocale_r() - get the locale of a program (MT-Safe)	setlocale(3C)
getlogin() - get name of user logged in on this terminal	getlogin(3C)
getlogin_r() - get name of user logged in and return name to buffer	getlogin(3C)
getmaxyx() - get additional cursor and window coordinates	getbegyx(3X)
getmntent() - get a file system description file entry	getmntent(3X)
getmsg() - receive next message from a STREAMS file	getmsg(2)
getnetbyaddr(): get network entry	getnetent(3N)
getnetbyname(): get network entry	getnetent(3N)
getnetconfig() - get network configuration data base entry	getnetconfig(3N)
getnetconfigent() - get network configuration data base entry	getnetconfig(3N)
getnetent(): get network entry	getnetent(3N)
getnetgrent() - get network group entry	getnetgrent(3C)
getnetname() - library routines for secure remote procedure calls	secure_rpc(3N)
getnetpath() - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
getnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
getn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
getopt - parse command options	getopt(1)
getopt(), optarg(), optind(), opterr() - get option letter from argument vector	getopt(3C)
getopts - parse utility (command) options	getopts(1)
getpagesize() - get the current page size	getpagesize(2)
getparyx() - get additional cursor and window coordinates	getbegyx(3X)
getpass() - read a password from terminal while suppressing echo	getpass(3C)
getpeername() - get address of connected peer	getpeername(2)

Description	Entry Name(Section)
getpgid() - get process group ID of specified process	getpid(2)
getpgrp() - get process group ID	getpid(2)
getpgrp2() - get process group ID of specified process	getpid(2)
getpid() - get process ID	getpid(2)
getpmsg() - receive next message from a STREAMS file in a priority order	getmsg(2)
getppid() - get parent process ID	getpid(2)
getprdfent() - manipulate system default database entry	getprdfent(3)
getprdfnam() - manipulate system default database entry	getprdfent(3)
getpriority() - get process priority	getpriority(2)
getprivgrp - get special attributes for group	getprivgrp(1)
getprivgrp() - get special attributes for group	getprivgrp(2)
getprotobyname() - get protocol entry	getprotoent(3N)
getprotobyname_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprotobynumber() - get protocol entry	getprotoent(3N)
getprotobynumber_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprotoent() - get protocol entry	getprotoent(3N)
getprotoent_r() - get protocol entry (thread-safe)	getprotoent(3N)
getprpwaid() - get protected password database audit ID	getprpwent(3)
getprpwnam() - get protected password database user name	getprpwent(3)
getprpwuid() - get protected password database user ID	getprpwent(3)
getprtcent() - manipulate terminal control database entry	getprtcent(3)
getprtcnam() - manipulate terminal control database entry	getprtcent(3)
getpublickey() - retrieve public or secret key	getpublickey(3M)
getpw() - get name from UID (obsolete)	getpw(3C)
getpwent() - get next password file entry	getpwent(3C)
getpwnam() - get password file entry matching login name name()	getpwent(3C)
getpwuid() - get password file entry matching uid()	getpwent(3C)
getrlimit() - get system resource consumption limit	getrlimit(2)
getrlimit64() - file system API to support large files	creat64(2)
getrpcbyname(): get RPC entry	getrpcent(3C)
getrpcbynumber(): get RPC entry	getrpcent(3C)
getrpcent(): get RPC entry	getrpcent(3C)
getrpcport() - get RPC port number	getrpcport(3N)
getrusage() - get information about resource utilization	getrusage(2)
gets(), fgets() - get a string from a standard input stream	gets(3S)
getsecretkey() - retrieve public or secret key	getpublickey(3M)
getservbyname() - get service entry	getservent(3N)
getservbyname_r() - get service entry (thread-safe)	getservent(3N)
getservbyport() - get service entry	getservent(3N)
getservbyport_r() - get service entry (thread-safe)	getservent(3N)
getservent() - get service entry	getservent(3N)
getservent_r() - get service entry (thread-safe)	getservent(3N)
getsid() - get session ID	getsid(2)
getsockname() - get socket address	getsockname(2)
getsockopt() - get options on sockets	getsockopt(2)
getspent() - get next secure password file entry	getspent(3C)
getspnam() - get secure password file entry matching name()	getspent(3C)
getspwaid() - get next secure password file audit ID	getspwent(3X)
getspwent() - get next secure password file entry	getspwent(3X)
getspwnam() - get secure password file entry matching login name name()	getspwent(3X)
getspwuid() - get secure password file entry matching uid()	getspwent(3X)
getstr() - get a multi-byte character string from the terminal	getstr(3X)
getsubopt() - parse suboptions from a string	getsubopt(3C)
gettimeofday() - get system clock date and time	gettimeofday(2)
gettimer() - get value of a per-process timer	gettimer(3C)
getttx() - create message files for use by	mkmsgs(1)
getttx() - read text string from message file	getttx(3C)
getty for 2-way line accessible to uucp	uugetty(1M)
getty - set terminal type, modes, speed, and line discipline	getty(1M)
getty to remote terminal, spawn (call terminal)	ct(1)
gettydefs - speed and terminal settings used by getty	gettydefs(4)

Description	Entry Name(Section)
getuid() - get real user ID	getuid(2)
getusershell() - get legal user shells	getusershell(3C)
getutent() - get pointer to next entry in a utmp() file	getut(3C)
getutent_r() - get pointer to next entry in a utmp() file	getut(3C)
getutid() - get pointer to entry matching id() in a utmp() file	getut(3C)
getutid_r() - get pointer to entry matching id() in a utmp() file	getut(3C)
getutline() - get pointer to entry matching line() in a utmp() file	getut(3C)
getutline_r() - get pointer to entry matching line() in a utmp() file	getut(3C)
getutxent() - get pointer to next entry in a utmpx() file	getutx(3C)
getutxid() - get pointer to entry matching id() in a utmpx() file	getutx(3C)
getutxline() - get pointer to entry matching line() in a utmpx() file	getutx(3C)
getw() - get data word (integer) from a stream file	getc(3S)
getwc() - get wide character from a stream file	getwc(3C)
getwchar() - get wide character from a stream file	getwc(3C)
getwchar_unlocked() - get wide character from a stream file	getwc(3C)
getwc_unlocked() - get wide character from a stream file	getwc(3C)
getwd() - get pathname of current working directory	getwd(3C)
getwin() - dump window to and reload window from a file	getwin(3X)
getw_unlocked() - get data word (integer) from a stream file	getc(3S)
getx25 - get X.25 line	getx25(1M)
getyx() - get cursor and window coordinates	getyx(3X)
get_expiration_time() - add a specific time interval to the current	get_expiration_time(3T)
get_myaddress() - obsolete library routines for RPC	rpc_soc(3N)
get_wch() - get a wide character from a terminal	get_wch(3X)
get_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
gfrt0.o, gcr0.o - C and Pascal execution startup routines	crt0(3)
glob - echo without '\\' escapes	csh(1)
glob() - file name generation function	glob(3C)
global kernel symbol, get information for	getksym(2)
global search path for dynamically loadable kernel modules, change	modpath(2)
globfree() - free space associated with file name generation function	glob(3C)
glossary - a description of common HP-UX terms	glossary(9)
gmtime(), gmtime_r() - convert date and time to Greenwich Mean Time	ctime(3C)
goto - continue execution on specified line	csh(1)
goto, save/restore stack environment for non-local	setjmp(3C)
GPIO: return status lines of GPIO card	gpio_get_status(3I)
GPIO: set control lines on GPIO card	gpio_set_ctl(3I)
gpio_get_status() - return status lines of GPIO card	gpio_get_status(3I)
gpio_set_ctl() - set control lines on GPIO card	gpio_set_ctl(3I)
gprof - display call graph execution profile data	gprof(1)
grant access to STREAMS slave pty	grantpt(3C)
grantpt() - grant access to STREAMS slave pty	grantpt(3C)
graph and display execution profile data	gprof(1)
graphic character	glossary(9)
grep - search a file for a pattern (compact algorithm)	grep(1)
grget - get group information	pwget(1)
group access list	glossary(9)
group access list: get group access list	getgroups(2)
group access list: initialize group access list	initgroups(3C)
group access list: set group access list	setgroups(2)
group and owner of a file, change	chown(2)
group and password hashing and caching daemon	pwgrd(1M)
group and password hashing and caching statistics	pwgr_stat(1M)
group and/or owner, change in access control list (ACL)	chownacl(3C)
group availability (LVM), set volume	vgchange(1M)
group configuration, restore volume	vgcfgrestore(1M)
group entry, network, get or set	getnetgrent(3C)
group file	login(1)
group file, check	pwck(1M)
group	glossary(9)
group - group access and identification file, grp.h	group(4)

Description	Entry Name(Section)
group ID for job control, set process	setpgid(2)
group ID	glossary(9)
group ID, create session and set process	setsid(2)
group ID, foreground process, get	tcgetpgrp(3C)
group ID, foreground process, set	tcsetpgrp(3C)
group ID, set process group ID	setpgrp(2)
group ID: get real or effective group ID	getuid(2)
group ID: set group ID	setuid(2)
group ID: set real, effective, and/or saved group or user IDs	setresuid(2)
group IDs and names, print	id(1)
group information, get (grget)	pwget(1)
group manipulation functions, NIS+	nis_groups(3N)
group memberships, show	groups(1)
group of file, change	chown(1)
group of processes, send signal	kill(2)
group owner: change for an NIS+ object	nischgrp(1)
group() file, get entry from	getgrent(3C)
group, add a new	groupadd(1M)
group, associate with certain privileges	setprivgrp(1M)
group, delete from the system	groupdel(1M)
group, get or set special attributes	getprivgrp(2)
group, get special attributes for	getprivgrp(1)
group, log in to a new	newgrp(1)
group, modify a	groupmod(1M)
group-access privileged values, format of	privgrp(4)
groupadd - add a new group to the system	groupadd(1M)
groupdel - delete a group from the system	groupdel(1M)
groupmod - modify a group on the system	groupmod(1M)
groups of programs - maintain, update, and regenerate	make(1)
groups - show group memberships	groups(1)
groups, list of in network	netgroup(4)
grpck - group file checker	pwck(1M)
gsignal() - raise a software signal	ssignal(3C)
gtty() - control terminal device (Bell Version 6 compatibility)	stty(2)
halfdelay() - control input character delay mode	halfdelay(3X)
HALGOL programs, execute	opx25(1M)
halt or start auditing system	audctl(2)
halt or start auditing system	audsys(1M)
halt system operation	shutdown(1M)
halt then reboot the system	reboot(1M)
handle special functions of HP 2640- and HP 2621-series terminals	hp(1)
hangups, run a command immune to	nohup(1)
hard disk, flexible disk, or cartridge tape media, initialize	mediainit(1)
hardware insert- and delete-character features, enable or disable use of	idcok(3X)
hardware machine model/series identification	model(4)
hash codes, convert 9-digit to or from text for spell checking	spell(1)
hash - remember command location in search path	sh-bourne(1)
hash search tables, manage	hsearch(3C)
hash value for ELF files, compute	elf_hash(3E)
hashcheck - convert spelling reference list words to 9-digit hash codes for spell	spell(1)
hashing and caching statistics, password and group	pwgr_stat(1M)
hashing and caching, password and group, daemon	pwgrd(1M)
hashing encryption on large strings, generate	bigcrypt(3C)
hashing encryption, generate	crypt(3C)
hashmake - convert text words to 9-digit hash codes for spell	spell(1)
hashstat - print hash table effectiveness statistics	csh(1)
hasmntopt() - search mount option field in file system description file	getmntent(3X)
has_colors() - color manipulation functions	can_change_color(3X)
has_ic() - query functions for terminal insert and delay capability	has_ic(3X)
has_il() - query functions for terminal insert and delay capability	has_ic(3X)
havedisk() - get performance data from remote kernel	rstat(3N)

Index

All Volumes

Description	Entry Name(Section)
hcreate() - allocate space for new hash search table	hsearch(3C)
hdestroy() - destroy existing hash search table	hsearch(3C)
head - get first few lines in a file	head(1)
header file for future applications	portal(5)
header file of macros for handling device numbers	mknod(5)
header file organization, description of HP-UX	stdsyms(5)
header files, C, generate	rpcgen(1)
header, retrieve ELF class-dependent object file header	elf_getehdr(3E)
help for SCCS commands	scshelp(1)
herror() - resolver routines	resolver(3N)
hexadecimal equivalents: ASCII character set	ascii(5)
hexadecimal file dump	od(1)
HFS file system administration command	fsadm_hfs(1M)
HFS file system debugger	fsdb_hfs(1M)
HFS file system disk blocks, report number of free	df_hfs(1M)
hfs file system quota consistency checker	quotacheck_nfs(1M)
HFS file system, construct a new	newfs_hfs(1M)
HFS file system, construct	mkfs_hfs(1M)
HFS file system, copy with compaction	dcopy(1M)
HFS file system, list file names and statistics	ff_hfs(1M)
HFS file system: tune an existing file system	tunefs(1M)
HFS file systems, mount and unmount	mount_hfs(1M)
hidden file	chmod(1)
hidden file	chmod(2)
hier - file system hierarchy	hier(5)
hierarchical directory	glossary(9)
hierarchy, directory, recursively descend a, executing a function	ftw(3C)
hierarchy, file system	hier(5)
high resolution sleep	nanosleep(2)
hil - HP-HIL device driver	hil(7)
hilkbd - HP-HIL cooked keyboard driver	hilkbd(7)
history and input editor for interactive program commands	ied(1)
history - Display event history list	csh(1)
hline() - draw lines from single-byte characters and renditions	hline(3X)
hline_set() - draw lines from complex characters and renditions	hline_set(3X)
hold signal upon receipt	sighold(2V)
hold signal upon receipt	sigset(3C)
home directory	glossary(9)
HOME environment variable	login(1)
home machine is not available, login when VHE	vhe_allog(1M)
host and network byte order, convert values between	byteorder(3N)
host cpu, set name of	sethostname(2)
host entry, get, set, or end network	gethostent(3N)
host is Network Information System server or map master, list which	ypwhich(1)
host name	glossary(9)
host name resolution description	hostname(5)
host names database	hosts(4)
host status of local machines (RPC version), show	rup(1)
host system, set or display name of current	hostname(1)
host table, translate to name server file format	hosts_to_named(1M)
host test packets, send	ping(1M)
host, current, get name of	gethostname(2)
host, remote, execute command on a	on(1)
host2netname() - library routines for secure remote procedure calls	secure_rpc(3N)
hostname - host name resolution description	hostname(5)
hostname - set or display name of current host system	hostname(1)
hosts - hosts name database	hosts(4)
hosts, compute shortest path and route between	pathalias(1)
hosts, remote, authorizing access on local host	hosts.equiv(4)
hosts.equiv file	login(1)
hosts.equiv - security files authorizing access by remote hosts and users on local host	hosts.equiv(4)

Description	Entry Name(Section)
hosts_to_named - translate host table to name server file format	hosts_to_named(1M)
how to order HP-UX manuals	manuals(5)
HP 2640- and HP 2621-series terminals, handle special functions of	hp(1)
HP 3000-mode packed decimal library	hppac(3X)
HP AdvanceLink and Basic Serial server	pcserver(1M)
HP Character-Terminal User Environment (CUE)	cue(1)
HP Distributed Printer System, advances a job to the top of a queue	pdpromote(1)
HP Distributed Printer System, deletes print objects	pddelete(1)
HP Distributed Printer System, enables paused objects to resume operation	pdresume(1)
HP Distributed Printer System, modifies attributes of submitted print jobs	pdmod(1)
HP Distributed Printer System, modifies attributes of submitted print jobs	pdset(1)
HP Distributed Printer System, pauses jobs, physical printers, servers, or queues	pdpause(1)
HP Distributed Printer System, queries and lists the status of one or more print jobs	pdq(1)
HP Distributed Printer System, removes print jobs	pdr(1)
HP Distributed Printer System, resubmits previously submitted print jobs	pdresubmit(1)
HP Distributed Printer System, stop servers	pdshutdown(1)
HP Distributed Printer System, submits print jobs	pdpr(1)
hp - handle special functions of HP 2640- and HP 2621-series terminals	hp(1)
HP-HIL - cooked keyboard driver	hilkbd(7)
HP-HIL: device driver	hil(7)
HP-IB I/O bus driver	hpib(7)
HP-IB: allow interface to enable SRQ line on HP-IB	hpib_rqst_srvc(3I)
HP-IB: change active controllers on HP-IB	hpib_pass_ctl(3I)
HP-IB: conduct a serial poll on HP-IB	hpib_spoll(3I)
HP-IB: control EOI mode for HP-IB file	hpib_eoi_ctl(3I)
HP-IB: control response to parallel poll on HP-IB	hpib_card_ppoll_resp(3I)
HP-IB: control the Remote Enable line on HP-IB	hpib_ren_ctl(3I)
HP-IB: perform I/O with an HP-IB channel from buffers	hpib_io(3I)
HP-IB: return status of HP-IB interface	hpib_bus_status(3I)
HP-IB: send command bytes over HP-IB	hpib_send_cmd(3I)
HP-IB: stop activity on specified HP-IB	hpib_abort(3I)
HP-IB: wait until a particular parallel poll value occurs	hpib_wait_on_ppoll(3I)
HP-IB: wait until the requested status condition becomes true	hpib_status_wait(3I)
HP-UX Audio	asecure(1M)
HP-UX bootstrap and installation utility	hpux(1M)
HP-UX files, verify internal revision numbers of	revck(1M)
HP-UX header file organization, description of	stdsyms(5)
HP-UX implementations, magic numbers for	magic(4)
HP-UX kernel or kernel module, build a bootable	mk_kernel(1M)
HP-UX machine identification	model(4)
HP-UX manuals, list of orderable	manuals(5)
HP-UX model, print name of current	model(1)
<i>HP-UX Reference manual</i> , introduction	Introduction(9)
HP-UX system, configure and build	config(1M)
HP-UX terms, glossary of	glossary(9)
HP-UX, introduction	Introduction(9)
hp2686a - laserjet filter	lpfilter(1)
hp2934a - character printer filter	lpfilter(1)
hp9000s200 - is processor an HP 9000 Series 200?	machid(1)
hp9000s300 - is processor an HP 9000 Series 300?	machid(1)
hp9000s500 - is processor an HP 9000 Series 500?	machid(1)
hp9000s800 - is processor an HP 9000 Series 800?	machid(1)
HPDPS gateway printers, configuration file	pdgwcfg.conf(4)
hpib - Hewlett-Packard Interface Bus driver	hpib(7)
hpib_abort() - stop activity on specified HP-IB	hpib_abort(3I)
hpib_bus_status() - return status of HP-IB interface	hpib_bus_status(3I)
hpib_card_ppoll_resp() - control response to parallel poll on HP-IB	hpib_card_ppoll_resp(3I)
hpib_eoi_ctl() - control EOI mode for HP-IB file	hpib_eoi_ctl(3I)
hpib_io() - perform I/O with an HP-IB channel from buffers	hpib_io(3I)
hpib_pass_ctl() - change active controllers on HP-IB	hpib_pass_ctl(3I)
hpib_ren_ctl() - control the Remote Enable line on HP-IB	hpib_ren_ctl(3I)

Index All Volumes

Description	Entry Name(Section)
hpib_rqst_srvce() - allow interface to enable SRQ line on HP-IB	hpib_rqst_srvce(3I)
hpib_send_cmnd() - send command bytes over HP-IB	hpib_send_cmnd(3I)
hpib_spoll() - conduct a serial poll on HP-IB	hpib_spoll(3I)
hpib_status_wait() - wait until the requested status condition becomes true	hpib_status_wait(3I)
hpib_wait_on_ppoll() - wait until a particular parallel poll value occurs	hpib_wait_on_ppoll(3I)
HPPAC*: HP 3000-mode packed decimal library	hppac(3X)
hpux - HP-UX bootstrap and installation utility	hpux(1M)
HPUX scheduling policy	rtsched(2)
hsearch() - hash table search routine	hsearch(3C)
htonl() - convert values between host and network byte order	byteorder(3N)
htons() - convert values between host and network byte order	byteorder(3N)
hyperbolic cosine function, inverse	acosh(3M)
hyperbolic cosine functions	cosh(3M)
hyperbolic sine function, inverse	asinh(3M)
hyperbolic sine functions	sinh(3M)
hyperbolic tangent function, inverse	atanh(3M)
hyperbolic tangent functions	tanh(3M)
hyphen - find hyphenated words	hyphen(1)
hyphenated words, find	hyphen(1)
hypot() - Euclidean distance function	hypot(3M)
hypotenuse of a right triangle	hypot(3M)
i-number, list path name corresponding to	ff_hfs(1M)
i-numbers, generate path names from	ncheck(1M)
I/O (NLIO) Subsystem, Native Language	nlio(5)
I/O card access information, network	lan(7)
I/O data path width (in bits), set	io_width_ctl(3I)
I/O device control commands, generic	ioctl(5)
I/O device drivers	(see special files)
I/O device file, list a	lssf(1M)
I/O device interrupt (fault) control	io_on_interrupt(3I)
I/O device, who is currently using	fuser(1M)
I/O interface, reset an	io_reset(3I)
I/O interface, unlock or lock an	io_lock(3I)
I/O interrupts for the associated eid(), disable or enable	io_interrupt_ctl(3I)
I/O multiplexing, synchronous	select(2)
I/O on kernel memory based on symbol name, perform	kmem(7)
I/O operations on a stream file, get or reposition pointer for	fseek(3S)
I/O operations, set time limit for	io_timeout_ctl(3I)
I/O pipe to or from a process, open or close	popen(3S)
I/O read termination character on special file, set up	io_eol_ctl(3I)
I/O read, determine how last terminated	io_get_term_reason(3I)
I/O redirection	glossary(9)
I/O statistics, report	iostat(1)
I/O subsystem, diagnostic interface to	diag0(7)
I/O system, scan the	ioscan(1M)
I/O to a stream file, buffered binary	fread(3S)
I/O with an HP-IB channel from buffers, perform	hpib_io(3I)
I/O, asynchronous synchronize	aio_fsync(2)
I/O, asynchronous write	aio_write(2)
I/O, asynchronous, error status	aio_error(2)
I/O, asynchronous, POSIX	aio(5)
I/O, asynchronous, start list of operations	lio_listio(2)
I/O, cancel asynchronous	aio_cancel(2)
I/O, control character device special file	ioctl(2)
I/O, faster tape	ftio(1)
I/O, POSIX asynchronous	aio(5)
I/O, read asynchronous	aio_read(2)
I/O, return asynchronous status	aio_return(2)
I/O, suspend for asynchronous completion	aio_suspend(2)
I/O, wait for asynchronous completion	aio_suspend(2)
I/O: GPIO card, return status lines of	gpio_get_status(3I)

Description	Entry Name(Section)
I/O: GPIO card, set control lines on	gpio_set_ctl(3I)
i4admin - administer LicensePower/iFOR licensing	i4admin(1M)
i4lmd - starts the license server	i4lmd(1M)
i4start - LicensePower/iFOR server start tool	i4start(1M)
i4stop - LicensePower/iFOR server stop tool	i4stop(1M)
i4target - print information about local LicensePower/iFOR target id	i4target(1M)
i4tv - verify Network License Servers are working	i4tv(1M)
IBM 3179G/3192G, 3270, 3777 terminal emulator	sna(1)
iconv - character code set conversion	iconv(1)
iconv translation tables to a readable format, dump	dmpxlt(1)
iconv translation tables, generate	genxlt(1)
iconv() - code set conversion routine, convert character	iconv(3C)
iconv_close() - code set conversion routine, deallocate conversion descriptor	iconv(3C)
iconv_open() - code set conversion routine, return conversion descriptor	iconv(3C)
ID for job control, set process group	setpgid(2)
id - print user and group IDs and names	id(1)
ID to file path, map device	devnm(3)
ID, create session and set process group	setsid(2)
ID, effective current user, print or display	whoami(1)
ID, foreground process group, get	tcgetpgrp(3C)
ID, foreground process group, set	tcsetpgrp(3C)
ID, get real or effective user or group	getuid(2)
ID, get session	getsid(2)
ID, get terminal session	tcgetsid(3C)
ID, obtain the thread ID for the calling thread	pthread_self(3T)
ID, set process group ID	setpgrp(2)
ID, set user or group	setuid(2)
idcok() - enable or disable use of hardware insert- and delete-character features	idcok(3X)
ident - identify files in Revision Control System	ident(1)
IDENT protocol server, TCP/IP	identd(1M)
identd - TCP/IP IDENT protocol server	identd(1M)
identification file, /etc/issue	issue(4)
identification information, SCCS, get from files	what(1)
identifier, create interprocess communication	ftok(3C)
identifier, get for the current host	gethostid(2)
identifiers, compare two thread identifiers	pthread_equal(3T)
identify files in Revision Control System	ident(1)
identify terminal type	ttytype(1)
identify the user of a particular TCP connection	idlookup(1)
idleok() - terminal output control functions	clearok(3X)
idlookup - identify the user of a particular TCP connection	idlookup(1)
IDs, print user and group	id(1)
IDs, processor, determined	pthread_processor_bind_np(3T)
IDs, set real and effective user IDs	setreuid(2)
ied - input editor and command history for interactive programs	ied(1)
IEEE-488 bus driver	hpib(7)
if - execute command if expression evaluates true	csch(1)
if - execute command if previous command returns exit status 0	ksh(1)
if - execute command if previous command returns exit status 0	sh-posix(1)
ifconfig - configure network interface parameters	ifconfig(1M)
ignorable signals mask, set current	sigsetmask(2)
ignore signal	sighold(2V)
ignore signal	sigset(3C)
ignore signals	sigblock(2)
ilogb() - unbiased exponent function	ilogb(3M)
image file, main memory and kernel memory	mem(7)
image	glossary(9)
immediate terminal refresh, enable/disable	immedok(3X)
immedok() - enable or disable immediate terminal refresh	immedok(3X)
implementation-specific constants	limits(5)
implementations, HP-UX, magic numbers for	magic(4)

Description

Entry Name(Section)

import an LVM volume group onto the system	vgimport(1M)
in-core state with its state on disk, synchronize a file's	fsync(2)
inbound zone transfer, ancillary agent	named-xfer(1M)
inch() - input a single-byte character and rendition from a window	inch(3X)
inchnstr() - input an array of single-byte characters and renditions from a window	inchnstr(3X)
inchstr() - input an array of single-byte characters and renditions from a window	inchstr(3X)
include and conditional instructions, process C language	c++(1)
incoming mailbox file, print out mail in the	prmail(1)
incoming messages from other users to terminal, deny or permit <i>write</i> (1)	mesg(1)
incomplete executable, prepare for faster program start-up	fastbind(1)
increase data segment space allocation	brk(2)
increase mirrors for LVM logical volume	lvextend(1M)
increase space for LVM logical volume	lvextend(1M)
incremental file system dump (for backups)	dump(1M)
incrementally restore file system	restore(1M)
index() - BSD portability string routine	string(3C)
index, generate permuted	ptx(1)
inet - Internet protocol family	inet(7F)
inetd configuration file	inetd.conf(4)
inetd - Internet services daemon	inetd(1M)
inetd optional security file	inetd.sec(4)
inetd.conf - configuration file for inetd	inetd.conf(4)
inetd.sec - optional inetd security file	inetd.sec(4)
inetsvcs.conf configuration file for secure internet services	inetsvcs.conf(4)
inetsvcs_sec - enable or disable secure internet services	inetsvcs_sec(1M)
inet_addr() - Internet address manipulation routines	inet(3N)
inet_lnaof() - Internet address manipulation routines	inet(3N)
inet_makeaddr() - Internet address manipulation routines	inet(3N)
inet_netof() - Internet address manipulation routines	inet(3N)
inet_network() - Internet address manipulation routines	inet(3N)
inet_ntoa() - Internet address manipulation routines	inet(3N)
inet_ntoa_r() - Internet address manipulation routines	inet(3N)
infinity, test for	isinf(3M)
info - diskless client configuration information file	info(4)
infocmp - compare or print out terminfo descriptions	infocmp(1M)
information about an NIS map, query an NIS server for	yppoll(1M)
information about LVM logical volumes, display	lvdisplay(1M)
information about physical volumes in LVM volume group, display	pvdisplay(1M)
information about resource utilization, get	getrusage(2)
information about top processes on system, display and update	top(1)
information about users on remote machines, return	rnusers(3N)
information file, diskless client configuration	info(4)
information file, driver, for insf , mksf , lssf	devices(4)
information file, LVM physical volume group	lvmpvg(4)
information file, Virtual Home Environment	vhe_list(4)
information for a dynamically loaded kernel module, get	modstat(2)
information for a global kernel symbol, get	getksym(2)
information for raster frame-buffer device access	framebuf(7)
information in password file; used by finger , change	chfn(1)
information name service: new version	nis+(1)
information on current terminal	cur_term(3X)
information on loaded module (program or shared library)	dlget(3C)
information on loaded module (program or shared library)	dlmodinfo(3C)
information server, remote user	fingerd(1M)
information, access exported file system	exportent(3N)
information, audit, display as requested by parameters	audisp(1M)
information, changes NIS	ypupdate(3C)
information, current user, look up	finger(1)
information, display software product	swlist(1M)
information, get locale-specific (NLS)	locale(1)
information, get Terminal Session Manager state	tsm.info(1)

Description	Entry Name(Section)
information, network I/O card access	lan(7)
information, NLS, about native languages	nl_langinfo(3C)
information, SCCS identification, get from files	what(1)
information, system paging space	swapinfo(1M)
init	glossary(9)
init - process control initialization	init(1M)
init process script	inittab(4)
initgroups() - initialize group access list	initgroups(3C)
initial system loader	isl(1M)
initialisation functions for screen	initscr(3X)
initialization routine called only once, threads	pthread_once(3T)
initialization utility, NIS+ client and server	nisinit(1M)
initialize a NIS+ domain	nissetup(1M)
initialize a thread attribute object	pthread_attr_init(3T)
initialize an unnamed semaphore	sem_init(2)
initialize and connect all system FDDI network interfaces	fddissetup(1M)
initialize and destroy a mutex	pthread_mutex_init(3T)
initialize FDDI interface	fddiinit(1M)
initialize group access list	initgroups(3C)
initialize hard disk, flexible disk, or cartridge tape media	mediainit(1)
initialize NIS+ credentials for NIS+ principals	nisclient(1M)
initialize or destroy a mutex attribute object	pthread_mutexattr_init(3T)
initialize or destroy a read-write lock	pthread_rwlock_init(3T)
initialize or destroy a read-write lock attribute object	pthread_rwlockattr_init(3T)
initialize or destroy a thread condition variable	pthread_cond_init(3T)
initialize or destroy a thread condition variable attributes object	pthread_condattr_init(3T)
initialize semaphore in mapped file or anonymous memory region	msem_init(2)
initialize terminal based on terminal type	tset(1)
initialize, manipulate, and test signal sets	sigsetops(3C)
initiate connection on a socket	connect(2)
initscr() - screen initialisation functions	initscr(3X)
initstate(), setstate(), random(), srandom() - generate a pseudorandom number	random(3M)
inittab - script for the init process	inittab(4)
init_color() - color manipulation functions	can_change_color(3X)
init_pair() - color manipulation functions	can_change_color(3X)
innetgr() - get network group entry	getnetgrent(3C)
innstr() - input a multi-byte character string from a window	innstr(3X)
innwstr() - input a string of wide characters from a window	innwstr(3X)
inode - format of an inode	inode(4)
inode generation numbers, install random	fsirand(1M)
inode	glossary(9)
inode number	glossary(9)
inode numbers for VxFS file system, generate pathnames from	ncheck_vxfs(1M)
inode, clear	clri(1M)
inode_vxfs - format of VxFS inode	inode_vxfs(4)
input a complex character and rendition from a window	in_wch(3X)
input a multi-byte character string from a window	innstr(3X)
input a single-byte character and rendition from a window	inch(3X)
input a string of wide characters from a window	innwstr(3X)
input an array of complex characters and renditions from a window	in_wchnstr(3X)
input an array of single-byte characters and renditions from a window	inchnstr(3X)
input character, control delay mode	halfdelay(3X)
input conversion, formatted read from stream file or character string	scanf(3S)
input conversion, formatted, to a varargs argument	vscanf(3S)
input editor and command history for interactive programs	ied(1)
input mode control functions	cbreak(3X)
input queue, push a character onto	ungetch(3X)
input script format and semantics, localedef-command	localedef(4)
input single line from user keyboard	line(1)
input stream, push character back into	ungetc(3S)
input stream, push wide character back into	ungetwc(3C)

Index

All Volumes

Description	Entry Name(Section)
input string from a standard input stream	gets(3S)
input wide string from a stream file	fgetws(3C)
input, control blocking on	notimeout(3X)
input, convert formatted, from a window	mvsanw(3X)
input, discard	flushinp(3X)
input, read a line from standard	read(1)
input/output functions for screen file	scr_dump(3X)
input/output to a stream file, buffered binary	fread(3S)
input/output, buffered, standard stream file package	stdio(3S)
insch() - insert a single-byte character and rendition into a window	insch(3X)
insdelln() - delete or insert lines into a window	insdelln(3X)
insert a complex character and rendition into a window	ins_wch(3X)
insert a multi-byte character into a window	insnstr(3X)
insert a single-byte character and rendition into a window	insch(3X)
insert a wide-character string into a window	ins_nwstr(3X)
insert and delay capability, for terminal	has_ic(3X)
insert calls to <i>catgets(3C)</i> based on <i>findstr(1)</i> output	insertmsg(1)
insert lines into a window	insertln(3X)
insert or delete lines into a window	insdelln(3X)
insert or remove an element in a queue	insque(3C)
insert- and delete-character features, hardware, enable or disable use of	idcok(3X)
insertln() - insert lines into a window	insertln(3X)
insertmsg - use <i>findstr(1)</i> output to insert calls to <i>catgets(3C)</i>	insertmsg(1)
insf - install special (device) files	insf(1M)
insnstr() - insert a multi-byte character into a window	insnstr(3X)
insque() - insert an element in a queue	insque(3C)
insnstr() - insert a multi-byte character into a window	insnstr(3X)
install - install new commands	install(1M)
install Network Information Service databases, build and	ypinit(1M)
install new elm aliases for user or system	newalias(1)
install object files in binary directories	cpset(1M)
install random inode generation numbers	fsirand(1M)
install Software Distributor agent on remote systems	pushAgent(1M)
install special (device) files	insf(1M)
install, monitor, create, distribute, and manage software	sd(5)
install, software products	swinstall(1M)
install, update, or remove boot programs from a disk device	mkboot(1M)
installation and bootstrap utility, HP-UX	hpux(1M)
installed software, configure, unconfigure, reconfigure	swconfig(1M)
instr() - input a multi-byte character string from a window	innstr(3X)
ins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
ins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
ins_wstr() - insert a wide-character string into a window	ins_nwstr(3X)
int, round to nearest functions	rint(3M)
integer absolute value, return	abs(3C)
integer data types	inttypes(5)
integer division and remainder	div(3C)
integer to base-64 ASCII string, convert long	a64l(3C)
integer, convert string to long	strtol(3C)
integer, convert wide character string to long	wcstol(3C)
integer, long, convert to string,	ltostr(3C)
integers, convert between 3-byte integers and long integers	l3tol(3C)
interactive mail message processing system	mailx(1)
interactive programs, input editor and command history for	ied(1)
interactively write (talk) to another user	write(1)
interface for stape, tape2 and tape1, magnetic tape	mt(7)
interface parameters, configure network	ifconfig(1M)
interface status, show FDDI	fddistat(1M)
interface status, show PCI FDDI	fddipciadmin(1M)
interface to the TELNET protocol, user	telnet(1)
interface, block mode terminal	blmode(7)

Description	Entry Name(Section)
interface, Centronics-compatible	cent(7)
interface, extended general terminal	termiox(7)
interface, GPIO: return status lines of GPIO card	gpio_get_status(3I)
interface, GPIO: set control lines on GPIO card	gpio_set_ctl(3I)
interface, HP-IB: allow interface to enable SRQ line on HP-IB	hpib_rqst_srvce(3I)
interface, HP-IB: change active controllers on HP-IB	hpib_pass_ctl(3I)
interface, HP-IB: conduct a serial poll on HP-IB	hpib_spoll(3I)
interface, HP-IB: control EOI mode for HP-IB file	hpib_eoi_ctl(3I)
interface, HP-IB: control response to parallel poll on HP-IB	hpib_card_ppoll_resp(3I)
interface, HP-IB: control the HP-IB interface Remote Enable line	hpib_ren_ctl(3I)
interface, HP-IB: HP-IB I/O bus driver	hpib(7)
interface, HP-IB: perform I/O with an HP-IB channel from buffers	hpib_io(3I)
interface, HP-IB: return status of HP-IB interface	hpib_bus_status(3I)
interface, HP-IB: send command bytes over HP-IB	hpib_send_cmnd(3I)
interface, HP-IB: stop activity on specified HP-IB	hpib_abort(3I)
interface, HP-IB: wait until a particular parallel poll value occurs	hpib_wait_on_ppoll(3I)
interface, HP-IB: wait until the requested status condition becomes true	hpib_status_wait(3I)
interface, initialize FDDI	fdiinit(1M)
interface, Network Information Service client	ypclnt(3C)
interface, reset an I/O	io_reset(3I)
interface, stop and reset FDDI	fdidstop(1M)
interface, terminal: general terminal interface	termio(7)
interface, terminal: process-controlling terminal device file	tty(7)
interface, terminal: system console interface special file	console(7)
interface, terminal: Version 6/PWB-compatible terminal interface	sttyv6(7)
interface, unlock or lock an I/O	io_lock(3I)
interfaces to terminfo database	del_curterm(3X)
interfaces, initialize and connect all system FDDI network	fdidsetup(1M)
interleave factor	glossary(9)
interleaved paging and swapping, add swap device for	swapon(2)
internal attributes, change program's	chatr(1)
internal consistency of Authentication database, check	authck(1M)
internal revision numbers of HP-UX files, verify	revck(1M)
Internal Terminal Emulator (ITE)	glossary(9)
Internal Terminal Emulator (ITE), load keyboard mapping	itemap(1M)
internationalization	glossary(9)
Internet address manipulation routines	inet(3N)
Internet Boot Protocol server	bootpd(1M)
Internet domain name server	named(1M)
Internet protocol family	inet(7F)
Internet Protocol, IP	IP(7P)
Internet services daemon	inetd(1M)
Internet Transmission Control Protocol, TCP	TCP(7P)
Internet user datagram protocol	UDP(7P)
Internet user name directory service	whois(1)
Internet, send mail over the	sendmail(1M)
interpret ASA carriage control characters	asa(1)
interpreter, command (shell) with C-like syntax	csh(1)
interpreter/compiler for modest-sized programs	bs(1)
interprocess channel, create an	pipe(2)
interprocess communication facilities, report status	ipcs(1)
interprocess communication identifier, create	ftok(3C)
Interprocess communications	socket(7)
interrupt (fault) conditions, define for I/O device	io_on_interrupt(3I)
interrupt functions, allowing signals to	siginterrupt(2)
interrupt signal	glossary(9)
interrupt, atomically release blocked signals and wait for	sigpause(2)
interrupt, enable or disable flush	intrflush(3X)
interrupts for the associated <code>eid()</code> , disable or enable I/O	io_interrupt_ctl(3I)
interval timer, set or get value of process	getitimer(2)
interval timer, set	ualarm(2)

Index

All Volumes

Description	Entry Name(Section)
interval, suspend execution for a time	sleep(1)
interval, suspend execution for an interval	usleep(2)
interval, suspend execution for	sleep(3C)
intrflush() - enable or disable flush on interrupt	intrflush(3X)
intrinsic	glossary(9)
introduction to access control lists	acl(5)
introduction to command utilities and application programs	intro(1)
introduction to device special files	intro(7)
introduction to file formats	intro(4)
introduction to general information section	intro(9)
introduction to HP-UX operating system and HP-UX Reference	Introduction(9)
introduction to miscellany	intro(5)
introduction to POSIX.1c threads	pthread(3T)
introduction to subroutines and libraries	intro(3)
introduction to system calls	intro(2)
introduction to system maintenance commands and application programs	intro(1M)
inttypes - basic integer data types	inttypes(5)
inv - make unprintable and non-ASCII characters in a file invisible	vis(1)
inverse hyperbolic cosine function	acosh(3M)
inverse hyperbolic sine function	asinh(3M)
inverse hyperbolic tangent function	atanh(3M)
inwstr() - input a string of wide characters from a window	innwstr(3X)
in_wch() - input a complex character and rendition from a window	in_wch(3X)
in_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
in_wchstr() - input an array of complex characters and renditions from a window	in_wchstr(3X)
ioconfig - ioconfig entry format	ioconfig(4)
ioctl commands: STREAMS	streamio(7)
ioctl - generic I/O device control commands	ioctl(5)
ioctl() - control character device special file	ioctl(2)
ioctl: STREAMS module for converting	timod(7)
ioinit - initialize I/O system	ioinit(1M)
ioinit - maintain consistency between data structures and /etc/ioconfig	ioinit(1M)
iomap - physical memory address mapping	iomap(7)
ioscan - scan the I/O system	ioscan(1M)
iostat - report I/O statistics	iostat(1)
IOT fault, generate an	abort(3C)
io_eol_ctl - set up I/O read termination character on special file	io_eol_ctl(3I)
io_get_term_reason() - determine how last read terminated	io_get_term_reason(3I)
io_interrupt_ctl() - enable/disable interrupts for the associated eid()	io_interrupt_ctl(3I)
io_lock() , io_unlock() - lock and unlock an I/O interface	io_lock(3I)
io_on_interrupt() - device I/O interrupt (fault) control	io_on_interrupt(3I)
io_reset() - reset an I/O interface	io_reset(3I)
io_speed_ctl() - inform system of required transfer speed	io_speed_ctl(3I)
io_timeout_ctl() - establish a time limit for I/O operations	io_timeout_ctl(3I)
io_unlock() - unlock an I/O interface	io_lock(3I)
io_width_ctl() - set width (in bits) of data path	io_width_ctl(3I)
IP - Internet Protocol	IP(7P)
IP Multicast - Internet Protocol	IP(7P)
IP multicast routing daemon	mROUTED(1M)
IP network tunnel driver	tun(4)
IP port, bind socket to a privileged	bindresvport(3N)
ipcrm - remove a message queue, semaphore set, or shared memory identifier	ipcrm(1)
ipcs - report status of interprocess communication facilities	ipcs(1)
ISA configuration tool	eisa_config(1M)
isalnum() - character is alphanumeric	ctype(3C)
isalpha() - character is alpha	ctype(3C)
isascii() - character is 7-bit ASCII code	ctype(3C)
ISASCII() - character is 7-bit ASCII code	wctype(3C)
isastream() - determine if file descriptor refers to STREAMS device or STREAMS-based pipe	isastream(3C)
isatty() - find name of a terminal	ttyname(3C)
isctrl() - character is a control character	ctype(3C)

Description	Entry Name(Section)
iscomsec - check if system has been converted to a trusted system	iscomsec(2)
isdigit() - character is a digit	ctype(3C)
isendwin() - determine whether a screen has been refreshed	isendwin(3X)
isfinite() - floating-point finiteness macro	isfinite(3M)
isgraph() - character is a visible character	ctype(3C)
isgreater() - floating-point comparison macro (>)	isgreater(3M)
isgreaterequal() - floating-point comparison macro (>=)	isgreaterequal(3M)
isinf() - test for infinity	isinf(3M)
isl - initial system loader	isl(1M)
isless() - floating-point comparison macro (<)	isless(3M)
islessequal() - floating-point comparison macro (<=)	islessequal(3M)
islessgreater() - floating-point comparison macro (<>)	islessgreater(3M)
islower() - character is lowercase	ctype(3C)
isnan() - floating-point test for NaN	isnan(3M)
isnormal() - floating-point test for normalized value	isnormal(3M)
isprint() - character is a printing character	ctype(3C)
ispunct() - character is punctuation	ctype(3C)
isspace() - character is whitespace	ctype(3C)
issue - /etc/issue identification file	issue(4)
issue a shell command	system(3S)
isunordered() - floating-point comparison macro (unordered)	isunordered(3M)
isupper() - character is uppercase	ctype(3C)
iswalnum() - character is alphanumeric	wctype(3C)
iswalpha() - character is alpha	wctype(3C)
iswcntrl() - character is a control character	wctype(3C)
iswctype() - character has property defined by prop	wctype(3C)
iswdigit() - character is a digit	wctype(3C)
iswgraph() - character is a visible character	wctype(3C)
iswlower() - character is lowercase	wctype(3C)
iswprint() - character is a printing character	wctype(3C)
iswpunct() - character is punctuation	wctype(3C)
iswspace() - character is whitespace	wctype(3C)
iswupper() - character is uppercase	wctype(3C)
iswxdigit() - character is a hexadecimal digit	wctype(3C)
isxdigit() - character is a hexadecimal digit	ctype(3C)
is_linetouched() - window refresh control functions	is_linetouched(3X)
is_wintouched() - window refresh control functions	is_linetouched(3X)
ITE (Internal Terminal Emulator)	glossary(9)
itemap - load a keyboard mapping into the Internal Terminal Emulator	itemap(1M)
j0() , j1() , jn() - Bessel functions of the first kind	j0(3M)
j1() - Bessel function	j0(3M)
jn() - Bessel function	j0(3M)
job control	glossary(9)
job control, set process group ID for	setpgid(2)
job control, uucp status inquiry and	uustat(1)
job execution daemon	cron(1M)
job file, prototype, for at	proto(4)
jobs - list active jobs	csh(1)
jobs - list active jobs	ksh(1)
jobs - list active jobs	sh-posix(1)
join corresponding lines of several files or subsequent lines of one file	paste(1)
join - relational database operator	join(1)
jukebox device drivers, SCSI	autochanger(7)
justify lines left or right for NLS printing	nljust(1)
keep track of remotely mounted file systems	mount(3N)
kermit - communication software for serial and network connections	kermit(1)
kernel capability, associate group with	setprivgrp(1M)
kernel data structures and /etc/ioconfig consistency	ioinit(1M)
kernel definition, display system	sysdef(1M)
kernel	glossary(9)
kernel memory and main memory image file	mem(7)

Index All Volumes

Description	Entry Name(Section)
kernel memory based on symbol name, perform I/O on	kmem(7)
kernel module administration	kmadmin(1M)
kernel module, add, delete, update	kminstall(1M)
kernel module, get information for a dynamically loaded	modstat(2)
kernel module, unload on demand	moduload(2)
kernel modules, change global search path for dynamically loadable	modpath(2)
kernel modules, load on demand	modload(2)
kernel modules, loadable, register or unregister, with the running kernel	kmmodreg(1M)
kernel modules, update	kmupdate(1M)
kernel statistics server	rstatd(1M)
kernel system file, create	create_sysfile(1M)
kernel, build a bootable HP-UX kernel or kernel modules	mk_kernel(1M)
kernel, global symbol, get information for	getksym(2)
kernel, loadable modules for running kernel during boot	loadmods(4)
kernel, master kernel configuration information	master(4)
kernel, remote, get performance data from	rstat(3N)
key server, storing private encryption keys	keyserv(1)
key, change user's secure RPC key	chkey(1)
key, decrypt and store secret	keylogin(1)
key, delete secret key stored with keyserv	keylogout(1)
key, generate a DES encryption	makekey(1)
key, get name of	keyname(3X)
keyboard driver, HP-HIL cooked	hilkbd(7)
keyboard mapping, loading into the Internal Terminal Emulator	itemap(1M)
keyboard, how to obtain control characters from	ascii(5)
keyboard, PS/2 device driver	ps2(7)
keyboard/display data order, convert file	forder(1)
keyenvoy(1M) - talk to the keyserv process	keyenvoy(1M)
keylogin - decrypt and store secret key	keylogin(1)
keylogout - delete secret key stored with keyserv	keylogout(1)
keyname() - get name of key	keyname(3X)
keypad() - enable/disable abbreviation of function keys	keypad(3X)
keys in Network Information Service map, print the values of selected	ypmatch(1)
keys, ppp encryption keys file format	ppp.Keys(4)
keys, update the public keys in a NIS+ directory object	nisupdkeys(1M)
keyserv process, talk to the	keyenvoy(1M)
keyserv - server for storing private encryption keys	keyserv(1)
keysh - context-sensitive softkey shell	keysh(1)
keysh softkey file format	softkeys(4)
keywords, find manual information by; print out a manual entry	man(1)
key_decryptsession() - library routines for secure remote procedure calls	secure_rpc(3N)
key_encryptsession() - library routines for secure remote procedure calls	secure_rpc(3N)
key_gendes() - library routines for secure remote procedure calls	secure_rpc(3N)
key_name() - get name of key	keyname(3X)
key_secretkey_is_set() - library routines for secure remote procedure calls	secure_rpc(3N)
key_setsecret() - library routines for secure remote procedure calls	secure_rpc(3N)
kill a directory	rmdir(1)
kill a file or directory	rm(1)
kill line character	eraseswchar(3X)
kill - send signal to process; terminate process	kill(1)
kill - send termination or specified signal to a process	csh(1)
kill - terminate job or process	ksh(1)
kill - terminate job or process	sh-posix(1)
kill() - send signal to process or group of processes	kill(2)
kill() system call, 4.2 BSD-compatible	bsdproc(3C)
kill() system call, 4.2 BSD-compatible	killpg(2)
killall - kill all active processes	killall(1M)
killchar() - single-byte line kill	erasechar(3X)
killpg() - 4.2 BSD-compatible kill() system call	killpg(2)
kills the sendmail daemon	killsm(1M)
killsm - kills the sendmail daemon	killsm(1M)

Description	Entry Name(Section)
killwchar() - current line kill character	eraseswchar(3X)
kmadmin - kernel module administration	kmadmin(1M)
kmem, kmem - kernel memory and main memory image file	mem(7)
kmem - perform I/O on kernel memory based on symbol name	kmem(7)
kminstall - add, delete, update a kernel module	kminstall(1M)
kmmodreg - register or unregister loadable kernel modules with the running kernel	kmmodreg(1M)
kmsystem - set, query configuration and loadable flags for a module	kmsystem(1M)
kmtune - query, set, or reset system parameter	kmtune(1M)
kmupdate - update default kernel files or specified kernel modules	kmupdate(1M)
known uucp systems, list names of	uucp(1)
ksh - Korn shell command programming language	ksh(1)
1 - list directory contents in stream output format	ls(1)
l3tol() - convert 3-byte integer to long integer	l3tol(3C)
l64a() - convert long integer to base-64 value ASCII string	a64l(3C)
l64a_r() - convert between long integer and base-64 ASCII string	a64l(3C)
label checking, copy file systems with	volcopy(1M)
label checking, copy file systems with	volcopy_hfs(1M)
label checking, VxFS file system	volcopy_vxfs(1M)
label, define for formatting routines	setlabel(3)
label, soft, functions	slk_attroff(3X)
labelit - copy file systems with label checking	volcopy(1M)
labelit - copy file systems with label checking	volcopy_hfs(1M)
labelit - label for VxFS file system	volcopy_vxfs(1M)
labs() - return long integer absolute value	abs(3C)
LAN administration	lanadmin(1M)
LAN connectivity, verify with link-level loopback	linkloop(1M)
LAN device configuration and status, display	lanscan(1M)
lan - network I/O card access information	lan(7)
LAN, log in on a remote system over	vt(1)
lanadmin - local area network administration	lanadmin(1M)
lang - description of supported languages	lang(5)
LANG	glossary(9)
langinfo - language information constants	langinfo(5)
Language I/O (NLIO) Subsystem, Native	nlcio(5)
language information constants	langinfo(5)
language macro processor	m4(1)
language, configure on multi-language systems	geocustoms(1M)
language: arbitrary-precision arithmetic language	bc(1)
language: C language preprocessor	cpre(1)
language: text pattern scanning and processing language	awk(1)
languages, description of supported	lang(5)
languages, NLS information about native (local)	nl_langinfo(3C)
lanscan - display LAN device configuration and status	lanscan(1M)
large argument list(s), construct, and execute command	xargs(1)
large files, find differences between	bdiff(1)
large letters, make posters in	banner(1)
large strings, generate hashing encryption on	bigcrypt(3C)
last commands executed, show in reverse order	lastcomm(1)
last I/O read, determine how terminated	io_get_term_reason(3I)
last - indicate last logins of users and ttys	last(1)
last locations of allocated regions in program	end(3C)
last login date, show for each user	acctsh(1M)
last logins of users and ttys, indicate	last(1)
last part of a file, get lines from	tail(1)
lastb - indicate last bad logins of users and ttys	last(1)
lastcomm - show last commands executed in reverse order	lastcomm(1)
lastlogin - show last login date for each user	acctsh(1M)
1c - list directory contents	ls(1)
lckpwwdf() - control access to /etc/passwd file	lckpwwdf(3C)
ld - link editor	ld(1)
ldd - list dynamic dependencies of executable files or shared libraries	ldd(1)

Index

All Volumes

Description	Entry Name(Section)
ldcvvt(), (_ldcvvt()) - convert long double to string	ldcvvt(3C)
ldexp - load exponent of a floating-point number	ldexp(3M)
ldfcvt(), (_ldfcvt()) - convert long double to string	ldcvvt(3C)
ldgcvt(), (_ldgcvt()) - convert long double to string	ldcvvt(3C)
ldiv() - long integer division and remainder	div(3C)
ldterm - STREAMS terminal line discipline module	ldterm(7)
ldterm, set and get EUC code widths for	eucset(1)
leave - remind you when you have to leave	leave(1)
leaveok() - terminal output control functions	clearok(3X)
left or right justify lines for NLS printing	nljust(1)
left-to-right text character sequence in each line of a file, reverse the	rev(1)
legal login shells, list of	shells(4)
legal user shells, get	getusershell(3C)
length limited string of single-byte characters and renditions to a window, add	addchnstr(3X)
length of string, find	string(3C)
length of wide string, find	wcstring(3C)
let - evaluate arithmetic expression	ksh(1)
let - evaluate arithmetic expression	sh-posix(1)
letters, make posters in large	banner(1)
lfind() - linear search and update	lsearch(3C)
lgamma(), lgamma_r(), gamma(), signgam() - log gamma function	lgamma(3M)
lgamma_r(), gamma(), signgam(), lgamma() - log gamma function	lgamma(3M)
libcrash - crash dump access library	libcrash(5)
libcrash error or warning message, print	cr_perror(3)
liblu62.a - IBM APPC (Advanced Program-to-Program Communications) API	see sna(1)
libraries and object files, generate single executable file from	ld(1)
libraries and subroutines, introduction	intro(3)
library and application versions, coordinate for ELF	elf_version(3E)
library file, link editor and assembler	a.out(4)
library	glossary(9)
library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
library routines for client side calls	rpc_clnt_calls(3N)
library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
library routines for creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
library routines for dealing with creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
library routines for external data representation	xdr(3N)
library routines for external data representation	xdr_admin(3N)
library routines for external data representation	xdr_complex(3N)
library routines for external data representation	xdr_simple(3N)
library routines for external data representation stream creation	xdr_create(3N)
library routines for registering servers, rpc	rpc_svc_reg(3N)
library routines for remote procedure calls	rpc(3N)
library routines for remote procedure calls, XDR	rpc_xdr(3N)
library routines for RPC bind service	rpcbind(3N)
library routines for RPC servers	rpc_svc_calls(3N)
library routines for RPC, obsolete	rpc_soc(3N)
library routines for secure remote procedure calls	secure_rpc(3N)
library routines for server side remote procedure call errors	rpc_svc_err(3N)
library routines for the creation of server handles, rpc	rpc_svc_create(3N)
library structure for transport function argument structures (X/OPEN TLI-XTI)	t_alloc(3)
library, audio	see audio(5)
library, packed decimal, HP 3000-mode	hppac(3X)
library: crash dump access library	libcrash(5)
library: object file access library	elf(3E)
license level of operating system, display	uname(1)
license level of operating system, get	uname(2)
license server, start	i4lmd(1M)
LicensePower/iFOR licensing	i4admin(1M)
LicensePower/iFOR server start tool	i4start(1M)
LicensePower/iFOR server stop tool	i4stop(1M)
licensing, LicensePower/iFOR	i4admin(1M)

Description	Entry Name(Section)
LIF directory, list contents of a	lifs(1)
LIF files, copy to or from	lifcp(1)
LIF files, list contents of a LIF directory	lifs(1)
LIF files, remove	lifrm(1)
LIF files, rename	lifrename(1)
LIF files, write LIF volume header on file	lifinit(1)
LIF	glossary(9)
lif - logical interchange format description	lif(4)
lifcp - copy to or from LIF files	lifcp(1)
lifinit - write LIF volume header on file	lifinit(1)
lifs - list contents of a LIF directory	lifs(1)
lifrename - rename LIF files	lifrename(1)
lifrm - remove a LIF file	lifrm(1)
limit for I/O operations, set time	io_timeout_ctl(3I)
limit - limit usage by current process	csh(1)
limit, get or set system resource consumption	getrlimit(2)
limits - implementation-specific constants	limits(5)
limits, disk usage	quota(5)
line connection, establish an out-bound terminal	dial(3C)
line control functions, tty	tccontrol(3C)
line control, asynchronous serial modem	modem(7)
line from standard input, read a	read(1)
line	glossary(9)
line kill character	eraseswchar(3X)
line kill character, single-byte	erasechar(3X)
line number and symbol information, strip from an object file	strip(1)
line numbering filter	nl(1)
line numbers, add in front of each line in a file	nl(1)
line on HP-IB, control the Remote Enable	hpib_ren_ctl(3I)
line printer daemon for LP requests from remote systems	rlpdaemon(1M)
line printer device files	lp(7)
line printer	see printer
line printer spooling system	see LP
line - read one line from user input	line(1)
line speed, datacomm, and terminal settings used by getty	gettydefs(4)
line update status functions	redrawwin(3X)
line, dedicated, reserve for a purpose	ripoffline(3X)
line, get X.25	getx25(1M)
line, single, input from user keyboard	line(1)
line, SRQ, on HP-IB, allow interface to enable	hpib_rqst_srvce(3I)
line-feeds, remove multiple from output	ssp(1)
line-feeds, reverse, and backspaces, remove from text	col(1)
line-oriented text editor	ed(1)
line-oriented text editor	ex(1)
line-printer or CRT output, format text file for	nroff(1)
linear table search with optional update	lsearch(3C)
lines common to two sorted files, reject/select	comm(1)
lines in a file, count	wc(1)
lines in a file, cut out (extract) selected fields of	cut(1)
lines in a file, report adjacent repeated	uniq(1)
LINES - number of lines on terminal screen	LINES(3X)
lines of GPIO card, return status	gpio_get_status(3I)
lines on GPIO card, set control	gpio_set_ctl(3I)
lines, delete or insert into a window	insdelln(3X)
lines, draw from complex characters and renditions	hline_set(3X)
lines, draw from single-byte characters and renditions	hline(3X)
lines, insert into a window	insertln(3X)
lines, justify left or right for NLS printing	nljust(1)
lines, long, fold for finite-width output device	fold(1)
lines, merge corresponding lines of several files or subsequent lines of one file	paste(1)
lines, number of, on terminal screen	LINES(3X)

Index

All Volumes

Description	Entry Name(Section)
lines, reduce multiple adjacent blank to single blank line	ssp(1)
lines, remove preprocessor	unifdef(1)
link count	glossary(9)
link directories using symbolic links	ln(1)
link editor and assembler output format	a.out(4)
link editor	ld(1)
link editor, find correct ordering of object code files for single pass	lorder(1)
link - execute link() system call without error checks	link(1M)
link existing file to new file name	ln(1)
link	glossary(9)
link NIS+ objects	nisln(1)
link to a file, make a symbolic	symlink(2)
link to root, primary swap, or dump volume, remove LVM logical volume	lvrmboot(1M)
link() - link additional name to an existing file	link(2)
link() system call, execute without error checks	link(1M)
link, file, symbolic (soft)	symlink(4)
link, symbolic, read value of	readlink(2)
link-level loopback to verify LAN connectivity	linkloop(1M)
linker	glossary(9)
linking process, diagnostic information for dynamic linking	dlerror(3C)
linkloop - verify LAN connectivity with link-level loopback	linkloop(1M)
lio_listio() - start a list of asynchronous I/O operations	lio_listo(2)
list a special (I/O device) file	lsxf(1M)
list access control lists (ACLs) of files	lsacl(1)
list access privileges for group	getprivgrp(1)
list access rights to file(s)	getaccess(1)
list characteristics of nodes on FDDI ring	fddinet(1M)
list contents of a LIF directory	lifs(1)
list contents of directory	ls(1)
list contents of DOS directories	dosls(1)
list current processes	ps(1)
list current system users	who(1)
list device drivers in the system	lsdev(1M)
list dynamic dependencies of executable files or shared libraries	ldd(1)
list entries in sendmail mail queue	sendmail(1M)
list file names and statistics for file system (generic)	ff(1M)
list file names and statistics for HFS file system	ff_hfs(1M)
list file names and statistics for VxFS file system	ff_vxfs(1M)
list first few lines in a file	head(1)
list lines from last part of a file	tail(1)
list names of known uucp systems	uucp(1)
list of allowed login shells	shells(4)
list of network groups	netgroup(4)
list of orderable HP-UX manuals	manuals(5)
list path name corresponding to i-number	ff_hfs(1M)
list the contents of an NIS+ directory	nisls(1)
list user crontab file	crontab(1)
list users currently on the system	users(1)
list which host is Network Information System server or map master	ypwhich(1)
list, get group access	getgroups(2)
list, initialize group access	initgroups(3C)
list, name, get entries from	nlist(3C)
list, print formatted output of a varargs argument	vprintf(3S)
list, set group access	setgroups(2)
listen for connections on a socket	listen(2)
listen() - listen for connections on a socket	listen(2)
lists owners of outgoing network connections	owners(1M)
lists selected attribute values	pdl(1)
lists, access control, introduction to	acl(5)
listusers - display user login data	listusers(1M)
11 - list directory contents in long format	ls(1)

Description	Entry Name(Section)
<code>llrint()</code> - round to nearest long long function	<code>llrint</code> (3M)
<code>llround()</code> - round to long long function	<code>llround</code> (3M)
<code>ln</code> - link files and directories	<code>ln</code> (1)
load exponent of a floating-point number	<code>ldexp</code> (3M)
load exponent of a radix-independent floating-point number	<code>scalb</code> (3M)
load exponent of a radix-independent floating-point number	<code>scalbn</code> (3M)
load kernel modules on demand	<code>modload</code> (2)
load module, retrieve name	<code>dlgetname</code> (3C)
load operating system	<code>boot</code> (1M)
load shared library	<code>shl_load</code> (3X)
loadable flags for a module, set and query	<code>kmsystem</code> (1M)
loadable kernel modules, register or unregister, with the running kernel	<code>kmmodreg</code> (1M)
loadable modules for running kernel during boot	<code>loadmods</code> (4)
loaded module (program or shared library)	<code>dlget</code> (3C)
loaded module (program or shared library)	<code>dlmodinfo</code> (3C)
loader, dynamic	<code>dld.sl</code> (64)
loader, initial system	<code>isl</code> (1M)
<code>loadmods</code> - loadable modules for running kernel during boot	<code>loadmods</code> (4)
local (native) languages, NLS information about	<code>nl_langinfo</code> (3C)
local area network administration	<code>lanadmin</code> (1M)
local communication domain protocol	<code>UNIX</code> (7P)
local customs	<code>glossary</code> (9)
local file system mount statistics	<code>rmtab</code> (4)
local file system, restore incrementally	<code>vxrestore</code> (1M)
local host, authorizing access from remote hosts and users	<code>hosts.equiv</code> (4)
local LicensePower/iFOR target id, print information	<code>i4target</code> (1M)
local machines (RPC version), show host status of	<code>rup</code> (1)
local machines, show status of	<code>ruptime</code> (1)
local machines, show who is logged in on	<code>rwho</code> (1)
local names, NIS+	<code>nis_local_names</code> (3N)
local network machines, determine who is logged in on	<code>rusers</code> (1)
local network packet routing, system support for	<code>routing</code> (7)
local node, transfer NIS database from NIS server to	<code>ypxfr</code> (1M)
local user, verify a remote user as a	<code>rcmd</code> (3N)
local, dump of file system	<code>vxdump</code> (1M)
locale environment, generate	<code>localedef</code> (1M)
<code>locale</code> - get locale-specific (NLS) information	<code>locale</code> (1)
locale of a program, get or set the	<code>setlocale</code> (3C)
locale, current, query numeric formatting conventions of	<code>localeconv</code> (3C)
locale-specific (NLS) information, get	<code>locale</code> (1)
<code>localeconv()</code> - query numeric formatting conventions of current locale	<code>localeconv</code> (3C)
<code>localedef</code> -command input script format and semantics	<code>localedef</code> (4)
<code>localedef</code> - generate a locale environment	<code>localedef</code> (1M)
localedef scripts, symbolic translation file for	<code>charmap</code> (4)
localization	<code>glossary</code> (9)
<code>localtime()</code> , <code>localtime_r()</code> - convert date and time to local timezone	<code>ctime</code> (3C)
locate a program file including aliases and paths	<code>which</code> (1)
locate source, binary, and/or manual files for program	<code>whereis</code> (1)
location functions, window cursor	<code>move</code> (3X)
location information about NIS+ servers	<code>nis_cachemgr</code> (1M)
location of byte in memory, find	<code>memory</code> (3C)
locations beyond allocated program regions, first	<code>end</code> (3C)
lock a POSIX semaphore	<code>sem_wait</code> (2)
lock a POSIX semaphore without blocking	<code>sem_wait</code> (2)
lock a semaphore	<code>msem_lock</code> (2)
lock access to /etc/passwd file	<code>lckpwwd</code> (3C)
lock daemon, network	<code>lockd</code> (1M)
lock or attempt to lock a read-write lock for reading	<code>pthread_rwlock_rdlock</code> (3T)
lock or attempt to lock a read-write lock for writing	<code>pthread_rwlock_wrlock</code> (3T)
lock or try to lock a mutex	<code>pthread_mutex_lock</code> (3T)
lock or unlock an I/O interface	<code>io_lock</code> (3I)

Description	Entry Name(Section)
lock process address space	mlockall(2)
lock process into memory after allocating data and stack space	datalock(3C)
lock process, text, or data in memory	plock(2)
lock - protect terminal from use by others	lock(1)
lock segment in memory	mlock(2)
lock terminal against use by others	lock(1)
lockd - network lock daemon	lockd(1M)
lockf() - provide semaphores and record locking on files	lockf(2)
lockf64() - file system API to support large files	creat64(2)
locking of streams within a multi-thread application, explicit	flockfile(3S)
locking on files, provide semaphores and record	lockf(2)
LOFS file system, mount	mount_lofs(1M)
log administration functions, NIS+	nis_ping(3N)
log driver, receive error messages from the STREAMS log driver	strerr(1M)
log driver: STREAMS	strlog(7)
log files, remove outdated STREAMS error log files	strclean(1M)
log gamma function	lgamma(3M)
log in on another system over LAN	vt(1)
log in to system	login(1)
log messages and other information about RCS files, print	rlog(1)
log of uucp and uux transactions, access summary	uucp(1)
log system messages	syslogd(1M)
log(), logf() - natural logarithm functions	log(3M)
log, display the contents of NIS+ transaction	nislog(1M)
log, error, collect system diagnostic messages to form	dmesg(1M)
log, mail, displays the last part of	mtail(1M)
log, system, control	syslog(3C)
log10(), log10f() - common logarithm functions	log10(3M)
log10f(), log10() - common logarithm functions	log10(3M)
log1p() - natural logarithm function	log1p(3M)
log2(), log2f() - logarithm base two functions	log2(3M)
log2f(), log2() - logarithm base two functions	log2(3M)
logarithm base ten functions	log10(3M)
logarithm base two functions	log2(3M)
logarithm function, natural	log1p(3M)
logarithm functions, natural	log(3M)
logb() - radix-independent exponent	logb(3M)
logf(), log() - natural logarithm functions	log(3M)
logfile, viewing and saving SAM	samlog_viewer
logged in on local machines, show who is	rwho(1)
logged in on local network machines, determine who is	rusers(1)
logged in users' accounting information file	utmpx(4)
logger - make entries in the system log	logger(1)
logging and tracing binary files, format	netfnt(1M)
logging and tracing, configure subsystem database	nettlconf(1M)
logging configuration file, network tracing and	nettlgen.conf(4)
logging, control network	nettl(1M)
logical interchange format description	lif(4)
Logical Interchange Format	glossary(9)
logical volume (LVM), decrease physical extents allocated to	lvreduce(1M)
logical volume (LVM), prepare to be root, swap, or dump volume	lvlnboot(1M)
logical volume (LVM), stripe, increase space, increase mirrors	lvextend(1M)
logical volume characteristics (LVM), change	lvchange(1M)
logical volume in LVM volume group, create	lvcreate(1M)
logical volume	lvm(7)
Logical Volume Manager (LVM)	lvm(7)
logical volume mirrors in LVM volume groups, synchronize stale	vgsync(1M)
logical volumes (LVM), display information about	lvdisplay(1M)
logical volumes (LVM), merge two into one logical volume	lvmerge(1M)
logical volumes (LVM), prepare root file system for migration from partitions to	lvmmigrate(1M)
logical volumes (LVM), synchronize stale mirrors in	lvsync(1M)

Description	Entry Name(Section)
logical volumes from LVM volume group, remove	lvremove(1M)
logical volumes, export an LVM volume group and its associated	vgexport(1M)
logical volumes, split mirrored LVM logical volume into two	lvsplit(1M)
login data, display	logins(1M)
login date, show last for each user	acctsh(1M)
login directory	glossary(9)
login directory	login(1)
login environment, shell script to set up user's	profile(4)
login	glossary(9)
login information for rexec() and ftp	netrc(4)
login name	login(1)
login name of the user, get character-string	cuserid(3S)
login name of user, obtain	logname(3C)
login name, change	su(1)
login name, get	logname(1)
login password in Network Information System (NIS), change	yppasswd(1)
login password, change	passwd(1)
login server, remote	rlogin(1M)
login shell, change default	chsh(1)
login shells, list of allowed	shells(4)
login - sign on; start terminal session	login(1)
login - terminate login shell	csh(1)
login when VHE home machine is not available	vhe_allog(1M)
login, add a user	useradd(1M)
login, delete a user	userdel(1M)
login, display data for user	listusers(1M)
login, modify a user	usermod(1M)
login, remote	rlogin(1)
login/logoff records, convert to per-session accounting records	acctcon(1M)
loggingroup file	login(1)
loggingroup - group access and identification file, grp.h	group(4)
logins - display login data	logins(1M)
logins of users and ttys, indicate last	last(1)
LOGNAME environment variable	login(1)
logname - get login name	logname(1)
logname() - return login name of user	logname(3C)
logout - terminate login shell	csh(1)
long double floating-point number to string, convert	ldcvt(3C)
long double-precision number, convert string to	strtold(3C)
long file names, convert a file system to allow	convertfs(1M)
long files, find differences between	bdiff(1)
long integer to base-64 ASCII string, convert	a64l(3C)
long integer to string, convert	ltostr(3C)
long integers and 3-byte integers, convert between	l3tol(3C)
long lines, fold for finite-width output device	fold(1)
long long, round function	llround(3M)
long long, round to nearest function	llrint(3M)
long, round function	lround(3M)
long, round to nearest function	lrint(3M)
longjmp() - restore stack environment after non-local goto	setjmp(3C)
longname() - get verbose description of current terminal	longname(3X)
look for LVM volume groups on physical volumes	vgscan(1M)
look up current user information	finger(1)
look up symbol in shared library	shl_load(3X)
lookup sources	service.switch(1M)
loopback, link-level, to verify LAN connectivity	linkloop(1M)
lorder - find ordering relation for an object code library	lorder(1)
lost+found directory for fsck(1M) , make a	mklost+found(1M)
lower-case, translate characters to	conv(3C)
lowercase, translate wide characters to	wconv(3C)
lp - line printer device files	lp(7)

Index

All Volumes

Description	Entry Name(Section)
lp - print requests on an LP printer	lp(1)
LP printer, allow or prevent queuing requests	accept(1M)
LP printer, print/alter/cancel requests	lp(1)
LP printers, enable/disable	enable(1)
LP requests, cancel from spooling queue on remote system	rcancel(1M)
LP requests: daemon for LP requests from remote systems	rlpdaemon(1M)
LP requests: print status information	lpstat(1)
LP requests: print status of requests sent to remote system for printing	rlpstat(1M)
LP requests: send LP request to a remote system	rlp(1M)
LP spooler performance analysis information, print	lpana(1M)
LP spooling requests, move to a specified destination	lpsched(1M)
LP spooling scheduler, start or stop	lpsched(1M)
LP spooling system, configure the	lpadmin(1M)
lpadmin - configure the LP spooling system	lpadmin(1M)
lpalt - alter requests on an LP printer	lp(1)
lpana - print LP spooler performance analysis information	lpana(1M)
lpfence - set LP spooling request priority fence	lpsched(1M)
lpfilter - filters invoked by lp interface scripts	lpfilter(1)
lpmove - move LP spooling requests to specified destination	lpsched(1M)
lprpp - laserjet filter	lpfilter(1)
lpsched - start the LP spooling request scheduler	lpsched(1M)
lpshut - stop the LP spooling request scheduler	lpsched(1M)
lpstat - print LP request status information	lpstat(1)
lrand48(), rand48() - generate long-integer pseudo-random numbers	drand48(3C)
lrint() - round to nearest long function	lrint(3M)
lround() - round to long function	lround(3M)
ls - list directory contents	ls(1)
lsacl - list access control lists (ACLs) of files	lsacl(1)
lsdev - list device drivers in the system	lsdev(1M)
lsearch(), lfind() - linear search and update	lsearch(3C)
lseek() - move read/write file pointer; seek	lseek(2)
lseek64() - file system API to support large files	creat64(2)
lsf - list directory contents, showing subdirectories and executable files	ls(1)
lsr - recursively list directory contents and subtrees	ls(1)
lssf - list a special (I/O device) file	lssf(1M)
lstat/fstat/stat system call, data returned by	stat(5)
lstat() - get symbolic link status	lstat(2)
lstat64() - file system API to support large files	creat64(2)
lsx - list directory contents, sort across page instead of down	ls(1)
ltoa() - convert long integer to ASCII decimal	ltostr(3C)
ltoa_r() - convert long integer to ASCII decimal (MT-Safe)	ltostr(3C)
ltol3() - convert long integer to 3-byte integer	ltol(3C)
ltostr() - convert long integer to string	ltostr(3C)
ltostr_r() - convert long integer to string (MT-Safe)	ltostr(3C)
lvchange - change LVM logical volume characteristics	lvchange(1M)
lvcreate - create logical volume in LVM volume group	lvcreate(1M)
lvdisplay - display information about LVM logical volumes	lvdisplay(1M)
lvextend - stripe, increase space, increase mirrors for LVM logical volume	lvextend(1M)
lvlnboot - prepare LVM logical volume to be root, swap, or dump volume	lvlnboot(1M)
LVM (Logical Volume Manager)	lvm(7)
LVM logical volume characteristics, change	lvchange(1M)
LVM logical volume link to root, primary swap, or dump volume, remove	lvrmboot(1M)
lvm - Logical Volume Manager (LVM)	lvm(7)
LVM logical volume, decrease physical extents allocated to	lvreduce(1M)
LVM logical volume, prepare to be root, swap, or dump volume	lvlnboot(1M)
LVM logical volume, stripe, increase space, increase mirrors	lvextend(1M)
LVM logical volumes, display information about	lvdisplay(1M)
LVM logical volumes, merge two into one logical volume	lvmerge(1M)
LVM logical volumes, prepare root file system for migration from partitions to	lvmmigrate(1M)
LVM logical volumes, synchronize stale mirrors in	lvsync(1M)
LVM physical volume group information file	lvmpvg(4)

Description	Entry Name(Section)
LVM physical volume to other physical volumes, move allocated physical extents from	pvmove(1M)
LVM setup	lvm(7)
LVM volume group and its associated logical volumes, export	vgexport(1M)
LVM volume group availability, set	vgchange(1M)
LVM volume group configuration backup file, create or update	vgcfbackup(1M)
LVM volume group definition, remove from the system	vgremove(1M)
LVM volume group, change characteristics and access path of physical volume in	pvchange(1M)
LVM volume group, check or repair a physical volume in	pvck(1M)
LVM volume group, create logical volume in	lvcreate(1M)
LVM volume group, create physical volume for use in	pvccreate(1M)
LVM volume group, create	vgcreate(1M)
LVM volume group, display information about physical volumes in	pvddisplay(1M)
LVM volume group, extend by adding physical volumes	vgextend(1M)
LVM volume group, import onto the system	vgimport(1M)
LVM volume group, reduce by removing physical volumes	vgreduce(1M)
LVM volume group, remove logical volumes from	lvremove(1M)
LVM volume groups, display information about	vgdisplay(1M)
LVM volume groups, scan physical volumes for	vgscan(1M)
LVM volume groups, synchronize stale logical volume mirrors	vgsync(1M)
LVM, split mirrored logical volume into two logical volumes	lvsplit(1M)
lvmerge - merge two LVM logical volumes into one logical volume	lvmerge(1M)
lvmmigrate - prepare root file system for migration from partitions to LVM logical volumes ..	lvmmigrate(1M)
lvmpvg - LVM physical volume group information file	lvmpvg(4)
lvreduce - decrease physical extents allocated to LVM logical volume	lvreduce(1M)
lvremove - remove logical volumes from LVM volume group	lvremove(1M)
lvrmboot - remove LVM logical volume link to root, primary swap, or dump volume	lvrmboot(1M)
lvsplit - split mirrored LVM logical volume into two logical volumes	lvsplit(1M)
lvsync - synchronize stale mirrors in LVM logical volumes	lvsync(1M)
lwpstat - show Fibre Channel Light Weight Protocol network status	lwpstat(1M)
m4 - macro processor	m4(1)
machid - provide truth value about processor type	machid(1)
machine identification, display	uname(1)
machine identification, get	uname(2)
machine information, change	setuname(1M)
machine is not available, login when VHE home	vhe_altlog(1M)
machine number, display	uname(1)
machine number, get	uname(2)
machine-dependent programming values and constants	values(5)
machines (RPC version), show host status of local	rup(1)
machines, determine who is logged in on local network	rusers(1)
machines, return information about users on remote	rnusers(3N)
machines, show status of local	runtime(1)
machines, show who is logged in on local	rwho(1)
machines, write to specified remote	rwall(3N)
macro package for formatting documents, MM	mm(5)
macro package for formatting manpages	man(5)
macro processor for C, Ratfor and other programming languages	m4(1)
macros and functions, floating-point environment	fenv(5)
macros for handling device numbers, header file of	mknod(5)
macros for handling variable argument lists	stdarg(5)
macros for handling variable argument lists	varargs(5)
macros, check or print documents formatted using mm	mm(1)
madvise() - advise system of process's expected paging behavior	madvise(2)
magic - magic numbers for HP-UX implementations	magic(4)
magic number	glossary(9)
magnetic tape dump and restore protocol module, remote	rmt(1M)
magnetic tape manipulating program	mt(1)
mail aliases file, rebuild database	newaliases(1M)
MAIL environment variable	login(1)
mail file	login(1)
mail folder, read mail from specified	readmail(1)

Index

All Volumes

Description	Entry Name(Section)
mail folders, summarize by subject and sender	mailfrom(1)
mail in the incoming mailbox file, print out	prmail(1)
mail interface, batch	fastmail(1)
mail log, displays the last part of	mtail(1M)
mail message processing system, interactive	mailx(1)
mail queue, list entries in sendmail	sendmail(1M)
mail queue, printing	mailq(1)
mail response: "I am not here" indication	vacation(1)
mail - send mail to users or read mail	mail(1)
mail traffic statistics, print	mailstats(1)
mail - who is mine from?	from(1)
mail, notify users of new	newmail(1)
mail, read from specified mail folder	readmail(1)
mail, screen-oriented interface	elm(1)
mail, send over the Internet	sendmail(1M)
mailbox file, print out mail in the incoming	prmail(1)
mailboxes, notify users of new mail in	newmail(1)
mailer, convert binary file to ASCII for transmission by	uuencode(1)
mailfrom - summarize mail folders by subject and sender	mailfrom(1)
mailq - prints the mail queue	mailq(1)
mailstats - print mail traffic statistics	mailstats(1)
mailx - interactive mail message processing system	mailx(1)
main memory allocator	malloc(3C)
main memory and kernel memory image file	mem(7)
main memory space allocation, control	malloc(3C)
main memory space usage, display	malloc(3C)
maintain, update, and regenerate groups of programs	make(1)
maintainer, archive and library, for portable archives	ar(1)
maintaining cache location information about NIS+ servers	nis_cachemgr(1M)
major and minor device pair: STREAMS driver	clone(7)
major number	glossary(9)
make a delta (change) to an SCCS file	delta(1)
make a directory file	mkdir(2)
make a directory	mkdir(1)
make a directory, special, or ordinary file	mknod(2)
make a DOS directory	dosmkdir(1)
make a FIFO special file	mkfifo(3C)
make a file descriptor, for ELF files	elf_begin(3E)
make a file system (generic)	mkfs(1M)
make a lost+found directory for fsck(1M)	mklost+found(1M)
make a makefile	mkmf(1)
make a name for a temporary file	mktemp(1)
make a Network Information System database	makedbm(1M)
make a new file system (generic)	newfs(1M)
make a new HFS file system	newfs_hfs(1M)
make a shell archive package	shar(1)
make a special (device) file	mksf(1M)
make a string pointer for ELF files	elf_strptr(3E)
make a symbolic link to a file	symlink(2)
make a unique (usually temporary) file name	mktemp(3C)
make an HFS file system	mkfs_hfs(1M)
make FIFO (named pipe) special files	mkfifo(1)
make - maintain, update, and regenerate groups of programs	make(1)
make posters in large letters	banner(1)
make unprintable and non-ASCII characters in a file visible or invisible	vis(1)
makecontext() - manipulate user contexts	makecontext(2)
makedbm - make a Network Information System database	makedbm(1M)
makefile, create a	mkmf(1)
makekey - generate a DES encryption key	makekey(1)
makemap - creates database maps for sendmail	makemap(1M)
mallinfo() - display memory space usage	malloc(3C)

Description	Entry Name(Section)
<code>malloc()</code> - allocate block of main memory	malloc(3C)
<code>mallot()</code> - control memory space allocation	malloc(3C)
<code>man(1)</code> , fix manual pages for faster viewing with	fixman(1M)
<code>man</code> - find manual information by keywords; print out a manual entry	man(1)
<code>man</code> - macros for formatting manpages	man(5)
manage a binary search tree	tsearch(3C)
manage hash search tables	hsearch(3C)
manage system database of automatically pushed STREAMS modules	autopush(1M)
manage the pathalias database, access and	uupath(1)
manage, monitor, create, distribute, and install software	sd(5)
management functions, for pad	newpad(3X)
management, signal (<code>sighold()</code> , <code>sigrelse()</code> , <code>sigignore()</code>)	sighold(2V)
management, signal (<code>sigset()</code> , <code>sighold()</code> , <code>sigrelse()</code> , <code>sigignore()</code> , <code>sigpause()</code>)	sigset(3C)
Manager state information, get Terminal Session	tsm.info(1)
manager, menu-driven system administration	sam(1M)
manager, shell layer	shl(1)
Manager, Terminal Session	tsm(1)
manipulate connect accounting records	fwtmp(1M)
manipulate crash dump data	crashutil(1M)
manipulate device assignment database entry	getdvagent(3)
manipulate disk quotas	quotactl(2)
manipulate flags for ELF files	elf_flag(3E)
manipulate magnetic tape drive	mt(1)
manipulate protected password database entry	getprpwent(3)
manipulate routing tables manually	route(1M)
manipulate section data for ELF files	elf_getdata(3E)
manipulate system default database entry	getprdfent(3)
manipulate terminal control database entry	getprtcent(3)
manipulate user contexts	makecontext(2)
manipulate, initialize, and test signal sets	sigsetops(3C)
manipulation functions, NIS+ groups	nis_groups(3N)
manipulation routines, Internet address	inet(3N)
manpage, macro package for formatting	man(5)
manpages, introduction	Introduction(9)
mantissa and exponent, split double-precision number into	frexp(3M)
manual entries, macro package for formatting	man(5)
manual entry files for given name, find location of	whereis(1)
manual entry, print out a; find manual information by keywords	man(1)
manual information by keywords, find; print out a manual entry	man(1)
manual pages, on-line, create the cat files for	catman(1M)
manually manipulate routing tables	route(1M)
manuals - current list of HP-UX documentation	manuals(5)
manuals, list of orderable HP-UX	manuals(5)
map device ID to file path	devnm(3)
map master, list which host is Network Information System server or	ypwhich(1)
map object into virtual memory	mmap(2)
map of ASCII character set	ascii(5)
map security file, NIS	securenets(4)
map stream pointer to file descriptor	fileno(3S)
map, print all values in a Network Information Service	ypcat(1)
map, print the values of selected keys in Network Information Service	ypmatch(1)
map, query an NIS server for information about an NIS	yppoll(1M)
<code>map-mbone</code> - multicast router connection mapper	map-mbone(1M)
mapped file or anonymous memory region, initialize semaphore in	msem_init(2)
mapped file or anonymous region, remove semaphore in	msem_remove(2)
mapped file, synchronize a	msync(2)
mapped region, unmap a	munmap(2)
mapper, multicast router connection	map-mbone(1M)
mapper, universal addresses to RPC program number	rpcbind(1M)
mapping access protections, modify memory	mprotect(2)
mapping definitions, memory	mman(5)

Description

Entry Name(Section)

mapping, physical memory address	iomap(7)
maps, creates database maps for sendmail	makemap(1M)
mark a thread as detached to reclaim its resources when it terminates	pthread_detach(3T)
mark differences between two files	diffmk(1)
mask for file creation, set and get permissions	umask(2)
mask for file-creation, set access permissions mode	umask(1)
mask, set current ignorable signals	sigsetmask(2)
mass storage media or device, test for recording integrity and proper operation	mediainit(1)
master and slave pty, STREAMS, unlocking	unlockpt(3C)
master driver, STREAMS	tels(7)
master kernel configuration information	master(4)
master - master kernel configuration information	master(4)
master pty driver, STREAMS	ptm(7)
master, list which host is Network Information System server or map	ypwhich(1)
match filename patterns	fnmatch(3C)
match routines for regular expressions	regexp(3X)
matching routines, regular expression	regcomp(3C)
math.h - floating-point environment macros and functions	fenv(5)
math: absolute value functions	fabs(3M)
math: arccosine functions (degrees)	acosd(3M)
math: arccosine functions	acos(3M)
math: arcsine functions (degrees)	asind(3M)
math: arcsine functions	asin(3M)
math: arctangent functions (degrees)	atand(3M)
math: arctangent functions	atan(3M)
math: arctangent-and-quadrant functions	atan2(3M)
math: arctangent-and-quadrant functions (degrees)	atan2d(3M)
math: base-2 exponential function	exp2(3M)
math: Bessel functions of the first kind	j0(3M)
math: Bessel functions of the second kind	y0(3M)
math: ceiling function	ceil(3M)
math: common logarithm functions	log10(3M)
math: copysign functions	copysign(3M)
math: cosine functions (degrees)	cosd(3M)
math: cosine functions	cos(3M)
math: cube root functions	cbrt(3M)
math: decompose floating-point number	modf(3M)
math: error function and complementary error function	erf(3M)
math: Euclidean distance (hypotenuse) function	hypot(3M)
math: exponential functions	exp(3M)
math: exponential functions	expm1(3M)
math: extract mantissa and exponent from double-precision number	frexp(3M)
math: floating-point classification macro	fpclassify(3M)
math: floating-point comparison macro (<)	isless(3M)
math: floating-point comparison macro (<=)	islessequal(3M)
math: floating-point comparison macro (< >)	islessgreater(3M)
math: floating-point comparison macro (>)	isgreater(3M)
math: floating-point comparison macro (>=)	isgreaterequal(3M)
math: floating-point comparison macro (unordered)	isunordered(3M)
math: floating-point finiteness macro	isfinite(3M)
math: floating-point sign-determination macro	signbit(3M)
math: floating-point test for infinity	isinf(3M)
math: floating-point test for NaN	isnan(3M)
math: floating-point test for normalized value	isnormal(3M)
math: floor function	floor(3M)
math: functions and constants	math(5)
math: hyperbolic cosine functions	cosh(3M)
math: hyperbolic sine functions	sinh(3M)
math: hyperbolic tangent functions	tanh(3M)
math: inverse hyperbolic cosine function	acosh(3M)
math: inverse hyperbolic sine function	asinh(3M)

Description	Entry Name(Section)
math: inverse hyperbolic tangent function	atanh(3M)
math: load exponent of a floating-point number	ldexp(3M)
math: load exponent of a radix-independent floating-point number	scalb(3M)
math: load exponent of a radix-independent floating-point number	scalbn(3M)
math: log gamma function	lgamma(3M)
math: logarithm base two functions	log2(3M)
math: maximum value function	fmax(3M)
math: minimum value function	fmin(3M)
math: natural logarithm function	log1p(3M)
math: natural logarithm functions	log(3M)
math: next representable floating-point value	nextafter(3M)
math: positive difference function	fdim(3M)
math: power functions	pow(3M)
math: pseudorandom number generation functions	random(3M)
math: radix-independent exponent	logb(3M)
math: remainder function	remainder(3M)
math: remainder function with quotient	remquo(3M)
math: remainder functions	fmod(3M)
math: round function	round(3M)
math: round to long function	lround(3M)
math: round to long long function	llround(3M)
math: round to nearest int functions	rint(3M)
math: round to nearest long function	lrint(3M)
math: round to nearest long long function	llrint(3M)
math: sine functions (degrees)	sind(3M)
math: sine functions	sin(3M)
math: square root functions	sqrt(3M)
math: string-to-NaN conversion function	nan(3M)
math: tangent functions (degrees)	tand(3M)
math: tangent functions	tan(3M)
math: truncation function	trunc(3M)
math: unbiased exponent function	ilogb(3M)
mathematical text, preprocess and format for nroff	neqn(1)
maximum value function	fmax(3M)
mblen() - number of bytes in the multibyte character	multibyte(3C)
mbstowcs() - convert sequence of multibyte characters	multibyte(3C)
mbtowc() - number of bytes in the multibyte character	multibyte(3C)
mc - media changer manipulation utility	mc(1M)
mcrt0.o, crt0.o - C and Pascal execution startup routines	crt0(3)
measure time used to execute a command	time(1)
measure, convert units of	units(1)
media changer device driver, SCSI	autochanger(7)
media changer manipulation utility	mc(1M)
mediainit - initialize hard disk, flexible disk, or cartridge tape media	mediainit(1)
mem, kmem - main memory and kernel memory image file	mem(7)
memberships, show group	groups(1)
memccpy() - copy bytes from memory to another memory location	memory(3C)
memchr() - find first occurrence of byte in memory area	memory(3C)
memcmp() - compare byte with memory contents	memory(3C)
memcpy() - copy bytes from memory to another memory location	memory(3C)
memmove() - move memory contents	memory(3C)
memory address mapping, physical	iomap(7)
memory allocator for main memory	malloc(3C)
memory control operations, shared	shmctl(2)
memory for library structure (X/OPEN TLI-XTI)	t_free(3)
memory image file, main memory and kernel	mem(7)
memory mapping access protections, modify	mprotect(2)
memory mapping definitions	mman(5)
memory operations - copy, compare, test for contents, or set contents to value	memory(3C)
memory region, initialize semaphore in mapped file or anonymous	msem_init(2)
memory segment, get shared	shmget(2)

Index

All Volumes

Description	Entry Name(Section)
memory statistics, report virtual	vmstat(1)
memory, copy a file into	copylist(3C)
memory, lock process address space in	mlockall(2)
memory, lock process into after allocating data and stack space	datalock(3C)
memory, lock process, text, data, stack, or shared library in	plock(2)
memory, lock segment in	mlock(2)
memory, map object into virtual	mmap(2)
memory, perform I/O on kernel memory based on symbol name	knmem(7)
memory, unlock segment	munlock(2)
memory, unlock virtual address space	munlockall(2)
memorymap() - display contents of memory allocator	malloc(3C)
memset() - set area in memory to contain a specified byte	memory(3C)
menu-driven system administration manager	sam(1M)
merge and/or sort files	sort(1)
merge corresponding lines of several files or subsequent lines of one file	paste(1)
merge or add total accounting files	acctmerge(1M)
merge RCS revisions	rcsmerge(1)
merge - three-way file merge	merge(1)
merge two LVM logical volumes into one logical volume	lvmerge(1M)
mesg - permit or deny <i>write(1)</i> messages from other users to terminal	mesg(1)
message catalog file, create for modification	findmsg(1)
message catalog file, extract messages from	findmsg(1)
message catalog file, generate a formatted	gencat(1)
message catalog for reading, close or open NLS	catopen(3C)
message catalog	glossary(9)
message catalog, set the default	setcat(3)
message catalogs, find strings for inclusion in	findstr(1)
message control operations	msgctl(2)
message file, read text string from	gettxt(3C)
message files, create for use by gettxt()	mkmsgs(1)
message from a socket, receive	recv(2)
message operation permissions	glossary(9)
message operations	msgop(2)
message processing system, interactive mail	mailx(1)
message queue descriptor, close	mq_close(2)
message queue identifier (mqsid)	glossary(9)
message queue identifier, remove	ipcrm(1)
message queue, create/open	mp_open(2)
message queue, get	msgget(2)
message queue, receive a message	mq_receive(2)
message queue, register/cancel a notification request with a message queue	mq_notify(2)
message queue, send a message	mq_send(2)
message queue, set the blocking status	mq_setattr(2)
message queue, unlink	mq_unlink(2)
message queues, report status	ipcs(1)
message transcription system	answer(1)
message, broadcast simultaneously to all users	wall(1M)
message, display in standard format	pfmt(3C)
message, NLS program, get an	catgets(3C)
message, print libcrash error or warning message	cr_perror(3)
message, send a message on a stream	putmsg(2)
message, send or receive message queue message	msgop(2)
message, send to a socket	send(2)
messages from other users to terminal, deny or permit <i>write(1)</i>	mesg(1)
messages from system, log	syslogd(1M)
messages to system log, send	logger(1)
messages, diagnostic, collect to form system error log	dmesg(1M)
messages, display NIS+ error	nis_error(3N)
messages, error, extract from C source into a file	mkstr(1)
messages, system error	perror(3C)
meta() - enable/disable meta-keys	meta(3X)

Description	Entry Name(Section)
meta-keys, enable/disable	meta(3X)
metacharacter	glossary(9)
metric system, convert units to or from	units(1)
mfrt0.o, frt0.o - FORTRAN execution startup routines	crt0(3)
migration from partitions to LVM logical volumes, prepare root file system for	lvmmigrate(1M)
MIME (Multipurpose Internet Mail Extensions)	elm(1)
minimum I/O data transfer rate, inform system of required	io_speed_ctl(3I)
minimum value function	fmin(3M)
minor and major device pair: STREAMS driver	clone(7)
minor number	glossary(9)
mirrored LVM logical volume, split into two logical volumes	lvsplit(1M)
mirrors for LVM logical volume, increase	lvextend(1M)
mirrors in LVM logical volumes, synchronize stale	lvsync(1M)
mirrors in LVM volume groups, synchronize stale logical volume	vgsync(1M)
miscellany, introduction to	intro(5)
mkboot - install, or update boot programs from a disk device	mkboot(1M)
mkdir - make a directory	mkdir(1)
mkdir() - make a directory file	mkdir(2)
mkdirp() - create directories in a path	mkdirp(3G)
mkfifo - make FIFO (named pipe) special files	mkfifo(1)
mkfifo() - make a FIFO special file	mkfifo(3C)
mkfs - construct a file system (generic)	mkfs(1M)
mkfs - construct an HFS file system	mkfs_hfs(1M)
mkfs - construct VxFS file system	mkfs_vxfs(1M)
mkfs_vxfs - construct VxFS file system	mkfs_vxfs(1M)
mklost+found - make a lost+found directory for fsck(1M)	mklost+found(1M)
mkmf - make a makefile	mkmf(1)
mkmsgs - create message files for use by gettext()	mkmsgs(1)
mknod - create special and FIFO files	mknod(1M)
mknod() - make directory, special, or ordinary file	mknod(2)
mknod.h - header file of macros for handling device numbers	mknod(5)
mkpdf - create Product Description File from an input	mkpdf(1M)
mksf - make a special (device) file	mksf(1M)
mkstr - extract error messages from C source into a file	mkstr(1)
mktemp - make a name for a temporary file	mktemp(1)
mktemp() - make a unique (temporary) file name	mktemp(3C)
mtime() - convert time into calendar time value	ctime(3C)
mktimer() - allocate a per-process timer	mktimer(3C)
mkupath, uupath - access and manage the pathalias database	uupath(1)
mk_kernel - build a bootable HP-UX kernel or kernel modules	mk_kernel(1M)
mlock() - lock segment of process address space in memory	mlock(2)
mlockall() - lock all process address space in memory	mlockall(2)
MM macro package for formatting documents	mm(5)
mm - print or check documents formatted with the mm macros	mm(1)
mm - the MM macro package for formatting documents	mm(5)
mmap() - map object into virtual memory	mmap(2)
mmap64() - file system API to support large files	creat64(2)
mnttab , establish mount table /etc/mnttab	setmnt(1M)
mnttab - mounted file system table	mnttab(4)
mode	glossary(9)
mode mask for file-creation, set access permissions	umask(1)
mode, change file access permissions	chmod(1)
mode, change file access permissions	chmod(2)
mode, EOI, for HP-IB file, control	hpib_eoi_ctl(3I)
mode, place system in single-user	shutdown(1M)
mode, set the cursor mode	curs_set(3X)
model - HP-UX machine identification	model(4)
model - print name of current HP-UX version	model(1)
model , print name of current HP-UX	model(1)
modem - asynchronous serial modem line control	modem(7)
modem capability database	terminfo(4)

Index

All Volumes

Description	Entry Name(Section)
modem line control, asynchronous serial	modem(7)
modest-sized programs, compiler/interpreter for	bs(1)
modf() - decompose floating-point number	modf(3M)
modification times, set or update file	utime(2)
modification, access, and/or change times of a file, update	touch(1)
modification, create message catalog file for	findmsg(1)
modifies attributes of submitted print jobs	pdmod(1)
modifies attributes of submitted print jobs	pdset(1)
modify a group on the system	groupmod(1M)
modify a user login on the system	usermod(1M)
modify and delete user credentials for an authentication service	pam_setcred(3)
modify and display variables in the stable storage	setboot(1M)
modify environment for command execution	env(1)
modify memory mapping access protections	mprotect(2)
modify or view Access Control Lists	swacl(1M)
modify selected characters	tr(1)
modify, add, or delete access control list entry	setaclentry(3C)
modify, delete, copy, add, or summarize file access control lists (ACLs)	chacl(1)
modifying Network Information Service passwd database, daemon for	yppasswdd(1M)
modload() - load kernel modules on demand	modload(2)
modpath() - change global search path for dynamically loadable kernel modules	modpath(2)
modstat() - get information for a dynamically loaded kernel module	modstat(2)
module, Emulation for STREAMS pty	ptem(7)
module, Packet Mode, for STREAM pty	pckt(7)
module, PAM user policy definition service	pam_updbe(5)
module, remote magnetic tape dump and restore protocol	rmt(1M)
modules, loadable, for running kernel during boot	loadmods(4)
moduload() - unload a kernel module on demand	moduload(2)
monacct - create periodic accounting summary files	acctsh(1M)
monetary value to string, convert	strfmon(3C)
monitor daemon, audit-overflow	audomon(1M)
monitor I/O conditions on multiple file descriptors	poll(2)
monitor OSPF gateways	ospf_monitor(1M)
monitor profile data, display	prof(1)
monitor uucp subnetwork activity	uusub(1M)
monitor() - prepare execution profile	monitor(3C)
monitor, create, distribute, install, and manage software	sd(5)
monitor, network status	statd(1M)
more - file perusal filter for crt viewing	more(1)
motd file	login(1)
mount a file system	vfsmount(2)
mount and unmount CD-ROM file systems	pfs_mount(1M)
mount and unmount VxFS file system	mount_vxfs(1M)
mount - mount a file system (generic)	mount(1M)
mount - mount an LOFS file system	mount_lofs(1M)
mount - mount CDFS file systems	mount_cdfs(1M)
mount - mount HFS file systems	mount_hfs(1M)
mount - mount NFS file systems	mount_nfs(1M)
mount - mount VxFS file system	mount_vxfs(1M)
mount multiple file systems	mountall(1M)
mount NFS file systems automatically	automount(1M)
mount request server, NFS	mountd(1M)
mount request server, PFS	pfs_mountd(1M)
mount statistics, local file system	rmtab(4)
mount table /etc/mnttab , establish	setmnt(1M)
mount to remote file system, perform Network File System	vhe_u_mnt(1M)
mount() - mount a file system	mount(2)
mountable file system	glossary(9)
mounta11 - mount multiple file systems	mountall(1M)
mountd : NFS mount request server	mountd(1M)
mounted file system statistics, get	ustat(2)

Description	Entry Name(Section)
mounted file system table	mnttab(4)
mounted file systems, keep track of remotely	mount(3N)
mounts, show all remote	showmount(1M)
mount_vxfs - mount and unmount VxFS file system	mount_vxfs(1M)
mouse, PS/2 device driver	ps2(7)
move a directory (requires super-user)	mvdir(1M)
move allocated physical extents from one LVM physical volume to other physical volumes	pvmove(1M)
move file to new location	mv(1)
move files between systems	ftp(1)
move LP spooling requests to a specified destination	lpsched(1M)
move magnetic tape forward or backward by files or records	mt(1)
move multiple files to another directory	mv(1)
move read/write file pointer; seek	lseek(2)
move window	mvwin(3X)
move() - window cursor location functions	move(3X)
movement commands cursor, output to the terminal	mvcur(3X)
mpctl() - multiprocessor control	mpctl(2)
mprotect - modify memory mapping access protections	mprotect(2)
mq_close() - close a message queue descriptor	mq_close(2)
mq_getattr() - get status information and attributes associated with a message queue	mq_getattr(2)
mq_notify() - register/cancel a notification request with a message queue	mq_notify(2)
mq_open() - create/open a message queue	mq_open(2)
mq_receive() - receive a message from a message queue	mq_receive(2)
mq_send() - send a message to a message queue	mq_send(2)
mq_setattr() - set the blocking status of a message queue associated with descriptor	mq_setattr(2)
mq_unlink() - unlink a message queue	mq_unlink(2)
rand48() , jrand48() - generate signed long-integer pseudo-random numbers	drand48(3C)
mrinfo - multicast routing configuration information tool	mrinfo(1)
mrouted - IP multicast routing daemon	mrouted(1M)
msem_init() - initialize semaphore in mapped file or anonymous memory region	msem_init(2)
msem_lock - lock a semaphore	msem_lock(2)
msem_remove - remove semaphore in mapped file or anonymous region	msem_remove(2)
msem_unlock - unlock a semaphore	msem_unlock(2)
msgctl() - message control operations	msgctl(2)
msgget() - get message queue	msgget(2)
msgrcv() - receive message from message queue	msgop(2)
msgsnd() - send message to message queue	msgop(2)
msqid (message queue identifier)	glossary(9)
msync - synchronize a mapped file	msync(2)
mt - magnetic tape interface for stape, tape1, and tape2	mt(7)
mt - magnetic tape manipulating program	mt(1)
mtab , mounted file systems table	pfs_fstab(5)
mtail - displays the last part of the mail log	mtail(1M)
multi-byte character length limited string, get from the terminal	getnstr(3X)
multi-byte character string, get from the terminal	getstr(3X)
multi-byte character string, input from a window	innstr(3X)
multi-byte character, insert into a window	insnstr(3X)
multi-byte characters, add a string of, without rendition to a window and advance cursor	addnstr(3X)
multi-language systems, configure system language	geocustoms(1M)
multi-thread application, explicit locking of streams within a	flockfile(3S)
multibyte characters and strings conversions	multibyte(3C)
multibyte() - multibyte characters and strings conversions	multibyte(3C)
multicast router connection mapper	map-mbone(1M)
multicast routing configuration information tool	mrinfo(1M)
multicast routing daemon, IP	mrouted(1M)
multiple adjacent blank lines, reduce to single blank line	ssp(1)
multiple file systems, mount and unmount	mountall(1M)
multiple files, split file into	csplit(1)
multiple line-feeds, remove from output	ssp(1)
multiple <i>n</i> -line pieces, split a file into	split(1)
multiplexing, synchronous I/O	select(2)

Index

All Volumes

Description	Entry Name(Section)
multiprocessor control	mpctl(2)
Multipurpose Internet Mail Extensions (MIME)	elm(1)
multithreaded process trace	ttrace(2)
multiuser state	glossary(9)
munlock() - unlock segment of process virtual address space	munlock(2)
munlockall() - unlock entire process virtual address space	munlockall(2)
munmap() - unmap a mapped region	munmap(2)
mutex attribute object, initialize or destroy	pthread_mutexattr_init(3T)
mutex, get and set the prioceiling of	pthread_mutex_getprioceiling(3T)
mutex, initialize and destroy	pthread_mutex_init(3T)
mutex, lock or try to lock	pthread_mutex_lock(3T)
mutex, unlock	pthread_mutex_unlock(3T)
mv - move or rename files and directories	mv(1)
mvaddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
mvaddchnstr() - add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
mvaddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
mvaddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvaddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvaddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvaddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvadd_wch() - add a complex character and rendition to a window	add_wch(3X)
mvadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvadd_wchstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvchgat() - change renditions of characters in a window	chgat(3X)
mvcur() - output cursor movement commands to the terminal	mvcur(3X)
mvdelch() - delete character from a window	delch(3X)
mvderwin() - define window coordinate transformation	mvderwin(3X)
mvdir - move a directory (requires super-user)	mvdir(1M)
mvgetch() - get a single-byte character from the terminal	getch(3X)
mvgetnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
mvgetn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvgetstr() - get a multi-byte character string from the terminal	getstr(3X)
mvget_wch() - get a wide character from a terminal	get_wch(3X)
mvget_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvhline() - draw lines from single-byte characters and renditions	hline(3X)
mvhline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvinch() - input a single-byte character and rendition from a window	inch(3X)
mvinnchnstr() - input an array of single-byte characters and renditions from a window	innchnstr(3X)
mvinchstr() - input an array of single-byte characters and renditions from a window	inchstr(3X)
mvinnstr() - input a multi-byte character string from a window	innstr(3X)
mvinnwstr() - input a string of wide characters from a window	innwstr(3X)
mvinsch() - insert a single-byte character and rendition into a window	insch(3X)
mvinsnstr() - insert a multi-byte character into a window	insnstr(3X)
mvinsstr() - insert a multi-byte character into a window	insnstr(3X)
mvinstr() - input a multi-byte character string from a window	innstr(3X)
mvins_wstr() - insert a wide-character string into a window	ins_wstr(3X)
mvins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
mvins_wstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvinwstr() - input a string of wide characters from a window	innwstr(3X)
mvin_wch() - input a complex character and rendition from a window	in_wch(3X)
mvin_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvin_wchstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvprintw() - print formatted output in window	mvprintw(3X)
mvscanw() - convert formatted input from a window	mvscanw(3X)
mvvline() - draw lines from single-byte characters and renditions	hline(3X)
mvvline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvwaddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
mvwaddchnstr() - add length limited string of single-byte characters and renditions to a window	

Description	Entry Name(Section)
.....	addchnstr(3X)
mvwaddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
mvwaddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvwaddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvwaddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
mvwaddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
mvwadd_wch() - add a complex character and rendition to a window	add_wch(3X)
mvwadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
mvwadd_wchstr() - add an array of complex characters and renditions to a window	add_wchstr(3X)
mvwchgat() - change renditions of characters in a window	chgat(3X)
mvwdelch() - delete character from a window	delch(3X)
mvwgetch() - get a single-byte character from the terminal	getch(3X)
mvwgetnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
mvwgetn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvwgetstr() - get a multi-byte character string from the terminal	getstr(3X)
mvwget_wch() - get a wide character from a terminal	get_wch(3X)
mvwget_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
mvwhline() - draw lines from single-byte characters and renditions	hline(3X)
mvwhline_set() - draw lines from complex characters and renditions	hline_set(3X)
mvwin() - move window	mvwin(3X)
mvwinch() - input a single-byte character and rendition from a window	inch(3X)
mvwinchnstr() - input an array of single-byte characters and renditions from a window	inchnstr(3X)
mvwinchstr() - input an array of single-byte characters and renditions from a window	inchnstr(3X)
mvwinstr() - input a multi-byte character string from a window	innstr(3X)
mvwinnwstr() - input a string of wide characters from a window	innwstr(3X)
mvwinsch() - insert a single-byte character and rendition into a window	insch(3X)
mvwinsnstr() - insert a multi-byte character into a window	insnstr(3X)
mvwinsstr() - insert a multi-byte character into a window	insnstr(3X)
mvwinstr() - input a multi-byte character string from a window	innstr(3X)
mvwins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvwins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
mvwins_wstr() - insert a wide-character string into a window	ins_nwstr(3X)
mvwinwstr() - input a string of wide characters from a window	innwstr(3X)
mvwin_wch() - input a complex character and rendition from a window	in_wch(3X)
mvwin_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
mvwin_wchstr() - input an array of complex characters and renditions from a window	in_wchstr(3X)
mvwprintw() - print formatted output in window	mvprintw(3X)
mvwscanw() - convert formatted input from a window	mvscanw(3X)
mvwvline() - draw lines from single-byte characters and renditions	hline(3X)
mvwvline_set() - draw lines from complex characters and renditions	hline_set(3X)
name database, network	networks(4)
name database, service	services(4)
name for a temporary file, make	mktemp(1)
name list (symbol table) of object code file, print	nm(1)
name of a file, change	mv(1)
name of a slave pty, get the	ptsname(3C)
name of current host system, set or display	hostname(1)
name of current HP-UX model, print	model(1)
name of current NIS domain, get or set	getdomainname(2)
name of device	devnm(1M)
name of file owner or group, change	chown(1)
name of key, get	keyname(3X)
name of operating system, display	uname(1)
name of operating system, get	uname(2)
name of the user's terminal or pseudo-terminal, get	tty(1)
name server file format, translate host table to	hosts_to_named(1M)
name server, domain, send signals to	sig_named(1M)
name server, Internet domain	named(1M)
name servers, query interactively	nslookup(1)

Index

All Volumes

Description

Entry Name(Section)

Name Service Switch backend libraries, query	nsquery(1)
name service: new version	nis+(1)
name space: attach a STREAMS file descriptor	fattach(3C)
name, change login	su(1)
name, get disk description by its	getdiskbyname(3C)
name, get login	logname(1)
name, print working directory	pwd(1)
name, set or display Network Information Service domain	domainname(1)
name, user, PAM routine to retrieve	pam_get_user(3)
name-service switch, configuration file for	nsswitch.conf(4)
name-to-address translation, generic transport	netdir(3N)
name: change the name of a file	rename(2)
name: create a name for a temporary file	tmpnam(3S)
name: detach a name from a STREAMS-based file descriptor	fdetach(3C)
name: find name of a terminal	ttname(3C)
name: get character-string representation of user login name	cuserid(3S)
name: get entries from name list	nlist(3C)
name: get name from UID (obsolete)	getpw(3C)
name: get name of current host	gethostname(2)
name: obtain user login name	logname(3C)
name: set the name of host cpu	sethostname(2)
named - Internet domain name server	named(1M)
named pipes, create	mknod(1M)
named-xfer - ancillary agent for inbound zone transfers	named-xfer(1M)
names and statistics for HFS file system, list file	ff_hfs(1M)
names database, host	hosts(4)
names from i-numbers, generate path	ncheck(1M)
names of known uucp systems, list	uucp(1)
names, NIS+ local names	nis_local_names(3N)
names, print user and group	id(1)
names: extract portions of path names	basename(1)
namespace functions, NIS+	nis_names(3N)
namespace: remove NIS+ objects	nismrm(1)
nan() - string-to-NaN conversion function	nan(3M)
NaN, test for	isnan(3M)
nanosleep() - high resolution sleep	nanosleep(2)
napms() - suspend the calling process	napms(3X)
native language	glossary(9)
Native Language I/O (NLIO) Subsystem	nlcio(5)
Native Language Support (NLS)	glossary(9)
native languages, NLS information about	nl_langinfo(3C)
natural logarithm function	log1p(3M)
natural logarithm functions	log(3M)
ncheck - generate path names from i-numbers	ncheck(1M)
ncheck - generate pathnames from inode numbers for VxFS file system	ncheck_vxfs(1M)
ncheck_vxfs - generate pathnames from inode numbers for VxFS file system	ncheck_vxfs(1M)
nc_perror() - get network configuration data base entry	getnetconfig(3N)
nc_spperror() - get network configuration data base entry	getnetconfig(3N)
ndd - network tuning	ndd(1M)
ndir.h - format of HP-UX directory streams	ndir(5)
nearbyint() - round to nearest int function	rint(3M)
neighboring systems description file format, ppp	ppp.Systems(4)
neqn - format mathematical text for nroff	neqn(1)
neqn , tbl , and nroff / troff constructs, remove	deroff(1)
netconfig - network configuration database	netconfig(4)
netdir() - generic transport name-to-address translation	netdir(3N)
netdir_free() - generic transport name-to-address translation	netdir(3N)
netdir_getbyaddr() - generic transport name-to-address translation	netdir(3N)
netdir_getbyname() - generic transport name-to-address translation	netdir(3N)
netdir_options() - generic transport name-to-address translation	netdir(3N)
netdir_perror() - generic transport name-to-address translation	netdir(3N)

Description	Entry Name(Section)
<code>netdir_spperror()</code> - generic transport name-to-address translation	netdir(3N)
<code>netfmt</code> - format tracing and logging binary files	netfmt(1M)
<code>netgroup</code> : list of network groups	netgroup(4)
<code>netname2host()</code> - library routines for secure remote procedure calls	secure_rpc(3N)
<code>netname2user()</code> - library routines for secure remote procedure calls	secure_rpc(3N)
NETPATH component, get /etc/netconfig entry corresponding to	getnetpath(3N)
<code>netrc</code> - login information for <code>rexec()</code> and <code>ftp</code>	netrc(4)
<code>netstat</code> - show network status	netstat(1)
<code>nettl</code> - control network tracing and logging	nettl(1M)
<code>nettlconf</code> - configure tracing and logging commands	nettlconf(1M)
<code>nettlgen.conf</code> - network tracing and logging configuration file	nettlgen.conf(4)
network and host byte order, convert values between	byteorder(3N)
network connections, outgoing, list owners of	owners(1M)
network entry, get or set	getnetent(3N)
Network File System (NFS) mount to remote file system, perform	vhe_u_mnt(1M)
network file system device files	nfs(7)
Network File System statistics	nfsstat(1M)
network group entry, get or set	getnetgrent(3C)
network groups, list of	netgroup(4)
network host entry, get, set, or end	gethostent(3N)
network I/O card access information	lan(7)
network information name service: new version	nis+(1)
Network Information Service (NIS) binder processes	ypserv(1M)
Network Information Service client interface	ypclnt(3C)
Network Information Service database and directory structure	ypfiles(4)
Network Information Service database, create or rebuild	ypmake(1M)
Network Information Service databases, build and install	ypinit(1M)
Network Information Service database, force propagation of a	yppush(1M)
Network Information Service domain name, set or display	domainname(1)
Network Information Service map, print all values in a	ypcat(1)
Network Information Service map, print the values of selected keys in	ypmatch(1)
Network Information Service <code>passwd</code> database, daemon for modifying	yppasswdd(1M)
Network Information Service server, bind to a particular	ypset(1M)
Network Information Service, update user password in	yppasswd(3N)
Network Information Service: NIS map, query an NIS server for information about an	yppoll(1M)
Network Information System (NIS), change login password in	yppasswd(1)
Network Information System database, make a	makedbm(1M)
Network Information System server or map master, list which host is	ypwhich(1)
network interface parameters, configure	ifconfig(1M)
network interfaces, initialize and connect all system FDDI	fddisetup(1M)
Network License Servers, verify working	i4tv(1M)
network lock daemon	lockd(1M)
network name database	networks(4)
network packet routing, system support for local	routing(7)
network <code>rwall</code> server	rwalld(1M)
network station address string conversion routines	net_aton(3C)
network status monitor	statd(1M)
network status show	netstat(1)
network status, show, Fibre Channel Light Weight Protocol	lwpstat(1M)
network test packets, send	ping(1M)
Network Time Protocol (NTP) daemon	xntpd(1M)
Network Time Protocol (NTP), query program	ntpq(1M)
Network Time Protocol (NTP), set time and date	ntpdate(1M)
network tools, FDDI	fddi(7)
network tracing and logging configuration file	nettlgen.conf(4)
network tracing and logging, control	nettl(1M)
network tuning	ndd(1M)
network tunnel driver, IP	tun(4)
network username server	rusersd(1M)
network, connect to the FDDI	fddiinit(1M)
network, dump of file system across	vxdump(1M)

Index All Volumes

Description	Entry Name(Section)
network, monitor uucp subnetwork activity	uusub(1M)
network, remote backup over	dump(1M)
network, restore file system incrementally across	restore(1M)
network, scatter data to check the	spray(3N)
network, write to all users over a	rwall(1M)
networks - network name database	networks(4)
net_aton() - network station address string conversion routines	net_aton(3C)
net_ntoa() - network station address string conversion routines	net_aton(3C)
new commands, install	install(1M)
new file system (generic), construct	newfs(1M)
new file, create	creat(2)
new HFS file system, construct a	newfs_hfs(1M)
new key, creating in publickey database file	newkey(1M)
new process, create a	fork(2)
new version of the network information name service	nis+(1)
new VxFS file system, construct a	newfs_vxfs(1M)
newalias - install new elm aliases for user or system	newalias(1)
newaliases - rebuilds the database for the mail aliases file	newaliases(1M)
newarray - configure a disk array	newarray(1M)
newform - change or reformat a text file	newform(1)
newfs - construct a new file system (generic)	newfs(1M)
newfs - construct a new HFS file system	newfs_hfs(1M)
newfs - construct new VxFS file system	newfs_vxfs(1M)
newfs_vxfs - construct new VxFS file system	newfs_vxfs(1M)
newgrp - equivalent to exec newgrp	csh(1)
newgrp - equivalent to exec newgrp	ksh(1)
newgrp - equivalent to exec newgrp	sh-bourne(1)
newgrp - equivalent to exec newgrp	sh-posix(1)
newgrp - switch to a new group	newgrp(1)
newkey - create a new key in the publickey database file	newkey(1M)
newline character	glossary(9)
newline translation, enable/disable	nl(3X)
newmail - notify users of new mail in mailboxes	newmail(1)
newpad() - pad management functions	newpad(3X)
news - print news items	news(1)
newterm() - screen initialisation functions	initscr(3X)
newwin() - window creation functions	newwin(3X)
next representable floating-point value	nextafter(3M)
nextafter() , nextafterf() - next representable floating-point value	nextafter(3M)
nextafterf() , nextafter() - next representable floating-point value	nextafter(3M)
nextkey() - get next key in database (old single-data-base version)	dbm(3X)
NFS clients, directories to export to	exports(4)
NFS clients, export and unexport directories to	exportfs(1M)
NFS daemons	nfsd(1M)
NFS file system disk blocks, report number of free	df_hfs(1M)
NFS file system, determine which processes are using	fuser(1M)
NFS file systems, automatically mount	automount(1M)
NFS file systems, mount and unmount	mount_nfs(1M)
NFS mount request server	mountd(1M)
nfs - network file system device files	nfs(7)
NFS statistics	nfsstat(1M)
nfsd - NFS daemon	nfsd(1M)
nfsstat - Network File System statistics	nfsstat(1M)
nftw() - walk a file tree executing a function	ftw(3C)
nftw2() - walk a file tree executing a function	ftw(3C)
nftw64() - file sysmmmaptem API to support large files	fgetpos64(2)
nice - alter command priority	csh(1)
nice - run a command at nondefault priority	nice(1)
nice value	renice(1)
nice() - change priority of a process	nice(2)
NIS (Network Information Service) binder processes	ypserv(1M)

Description	Entry Name(Section)
NIS (Network Information System), change login password in	yppasswd(1)
NIS database from NIS server to local node, transfer	ypxfr(1M)
NIS domain name, set or display	domainname(1)
NIS domain, get or set name of current	getdomainname(2)
NIS information, changes	ypupdate(3C)
NIS information, server for changing	ypupdated(1M)
NIS map security file	securenets(4)
NIS map, query an NIS server for information about an	yppoll(1M)
NIS map, updates to	udppublickey(1M)
NIS maps or /etc files: creating NIS+ tables	nisaddent(1M)
NIS server for information about an NIS map, query an	yppoll(1M)
NIS server to local node, transfer NIS database from	ypxfr(1M)
NIS updating, configuration file for	updaters(1M)
NIS+ client and server initialization utility	nisinit(1M)
NIS+ credentials, initialize for NIS+ principals	nisclient(1M)
NIS+ credentials: creating	nisaddcred(1M)
NIS+ database access functions	nis_db(3N)
NIS+ database files and directory structure	nisfiles(4)
NIS+ default values: display	nisdefaults(1)
NIS+ directories: creating	nismkdir(1)
NIS+ directories: list the contents	nisls(1)
NIS+ directories: removing	nismkdir(1)
NIS+ directory object, update the public keys	nisupdkeys(1M)
NIS+ domain, initialize	nissetup(1M)
NIS+ domain, populate NIS+ tables	nispopulate(1M)
NIS+ error messages, display	nis_error(3N)
NIS+ error messages: displaying	niserror(1)
NIS+ functions, misc	nis_servers(3N)
NIS+ group manipulation functions	nis_groups(3N)
NIS+ groups: administering	nisgrpadm(1)
NIS+ local names	nis_local_names(3N)
NIS+ log administration functions	nis_ping(3N)
NIS+ namespace functions	nis_names(3N)
NIS+ namespace: return the state using a conditional expression	nistest(1)
nis+ - new version of the network information name service	nis+(1)
NIS+ object formats	nis_objects(3N)
NIS+ object, change owner	nischown(1)
NIS+ object: change access rights	nischmod(1)
NIS+ object: change the group owner	nischgrp(1)
NIS+ object: change the time to live value	nischt(1)
NIS+ object: remove from the namespace	nismrm(1)
NIS+ object: symbolically link	nisln(1)
NIS+ password information: changing	nispasswd(1)
nis+ password table	ttsyncd(1M)
NIS+ password update daemon	rpc.nispasswdd(1M)
NIS+ principals, initialize NIS+ credentials for	nisclient(1M)
NIS+ server statistics, report	nisstat(1M)
NIS+ servers, cache location information	nis_cachemgr(1M)
NIS+ servers, send ping to	nisping(1M)
NIS+ servers, set up	nisserver(1M)
NIS+ service daemon	rpc.nisd(1M)
NIS+ subroutines	nis_subr(3N)
NIS+ table functions	nis_tables(3N)
NIS+ tables and objects: displaying	niscat(1)
NIS+ tables in a NIS+ domain, populate	nispopulate(1M)
NIS+ tables: administering	nistbladm(1)
NIS+ tables: creating from corresponding /etc files or NIS maps	nisaddent(1M)
NIS+ tables: utility for searching	nismatch(1)
NIS+ transaction log, display contents	nislog(1M)
nis+ trusted table	ttsyncd(1M)
NIS+ utility to print out the contents of the shared cache file	nisshowcache(1M)

Description	Entry Name(Section)
nisaddcred - create NIS+ credentials	nisaddcred(1M)
nisaddent - create NIS+ tables from corresponding /etc files or NIS maps	nisaddent(1M)
niscat - display NIS+ tables and objects	niscat(1)
nischgrp - change the group owner of an NIS+ object	nischgrp(1)
nischmod - change access rights on an NIS+ object	nischmod(1)
nischown - change owner of an NIS+ object	nischown(1)
nischttl - change the time to live value of an NIS+ object	nischttl(1)
nisclient - initialize NIS+ credentials for NIS+ principals	nisclient(1M)
nisd - NIS+ service daemon	rpc.nisd(1M)
nisdefaults - display NIS+ default values	nisdefaults(1)
nisd_resolv - NIS+ service daemon	rpc.nisd(1M)
niserror - display NIS+ error messages	niserror(1)
nisfiles - NIS+ database files and directory structure	nisfiles(4)
nisgrep - utility for searching NIS+ tables	nismatch(1)
nisgrpadm - administer NIS+ groups	nisgrpadm(1)
nisinit - NIS+ client and server initialization utility	nisinit(1M)
nisln - symbolically link NIS+ objects	nisln(1)
nislog - display the contents of the NIS+ transaction log	nislog(1M)
nisls - list the contents of an NIS+ directory	nisls(1)
nismatch - utility for searching NIS+ tables	nismatch(1)
nismkdir - create NIS+ directories	nismkdir(1)
nispasswd - change NIS+ password information	nispasswd(1)
nispasswdd() - NIS+ password update daemon	rpc.nispasswdd(1M)
nisping - send ping to NIS+ servers	nisping(1M)
nispopulate - populate the NIS+ tables in a NIS+ domain	nispopulate(1M)
nisrm - remove NIS+ objects from the namespace	nisrm(1)
nisrmdir - remove NIS+ directories	nisrmdir(1)
nisserver - set up NIS+ servers	nisserver(1M)
nissetup - initialize a NIS+ domain	nissetup(1M)
nisshowcache - NIS+ utility to print out the contents of the shared cache file	nisshowcache(1M)
nisstat - report NIS+ server statistics	nisstat(1M)
nistbladm - administer NIS+ tables	nistbladm(1)
nistest - return the state of the NIS+ namespace using a conditional expression	nitest(1)
nisupdkeys - update the public keys in a NIS+ directory object	nisupdkeys(1M)
nis_add() - NIS+ namespace functions	nis_names(3N)
nis_addmember() - NIS+ group manipulation functions	nis_groups(3N)
nis_add_entry() - NIS+ table functions	nis_tables(3N)
nis_cachemgr - cache location information about NIS+ servers	nis_cachemgr(1M)
nis_checkpoint() - NIS+ log administration functions	nis_ping(3N)
nis_clone_object() - NIS+ subroutines	nis_subr(3N)
nis_creategroup() - NIS+ group manipulation functions	nis_groups(3N)
nis_db() - NIS+ database access functions	nis_db(3N)
nis_destroygroup() - NIS+ group manipulation functions	nis_groups(3N)
nis_destroy_object() - NIS+ subroutines	nis_subr(3N)
nis_dir_cmp() - NIS+ subroutines	nis_subr(3N)
nis_domain_of() - NIS+ subroutines	nis_subr(3N)
nis_error() - display NIS+ error messages	nis_error(3N)
nis_first_entry() - NIS+ table functions	nis_tables(3N)
nis_freenames() - NIS+ subroutines	nis_subr(3N)
nis_freeresult() - NIS+ namespace functions	nis_names(3N)
nis_freeservlist() - frees the list of NIS+ servers	nis_server(3N)
nis_freetags() - frees the NIS+ tags structure	nis_server(3N)
nis_getnames() - NIS+ subroutines	nis_subr(3N)
nis_getservlist() - return a list of NIS+ server structures	nis_server(3N)
nis_groups() - NIS+ group manipulation functions	nis_groups(3N)
nis_ismember() - NIS+ group manipulation functions	nis_groups(3N)
nis_leaf_of() - NIS+ subroutines	nis_subr(3N)
nis_lerror() - display NIS+ error messages	nis_error(3N)
nis_list() - NIS+ table functions	nis_tables(3N)
nis_local_directory() - NIS+ local names	nis_local_names(3N)
nis_local_group() - NIS+ local names	nis_local_names(3N)

Description	Entry Name(Section)
<code>nis_local_host()</code> - NIS+ local names	<code>nis_local_names</code> (3N)
<code>nis_local_names()</code> - NIS+ local names	<code>nis_local_names</code> (3N)
<code>nis_local_principal()</code> - NIS+ local names	<code>nis_local_names</code> (3N)
<code>nis_lookup()</code> - NIS+ namespace functions	<code>nis_names</code> (3N)
<code>nis_map_group()</code> - NIS+ group manipulation functions	<code>nis_groups</code> (3N)
<code>nis_mkdir()</code> - create databases to support NIS+ services	<code>nis_server</code> (3N)
<code>nis_modify()</code> - NIS+ namespace functions	<code>nis_names</code> (3N)
<code>nis_modify_entry()</code> - NIS+ table functions	<code>nis_tables</code> (3N)
<code>nis_names()</code> - NIS+ namespace functions	<code>nis_names</code> (3N)
<code>nis_name_of()</code> - NIS+ subroutines	<code>nis_subr</code> (3N)
<code>nis_next_entry()</code> - NIS+ table functions	<code>nis_tables</code> (3N)
<code>nis_objects()</code> - NIS+ object formats	<code>nis_objects</code> (3N)
<code>nis_perror()</code> - display NIS+ error messages	<code>nis_error</code> (3N)
<code>nis_ping()</code> - NIS+ log administration functions	<code>nis_ping</code> (3N)
<code>nis_print_group_entry()</code> - NIS+ group manipulation functions	<code>nis_groups</code> (3N)
<code>nis_print_object()</code> - NIS+ subroutines	<code>nis_subr</code> (3N)
<code>nis_remove()</code> - NIS+ namespace functions	<code>nis_names</code> (3N)
<code>nis_remove_member()</code> - NIS+ group manipulation functions	<code>nis_groups</code> (3N)
<code>nis_remove_entry()</code> - NIS+ table functions	<code>nis_tables</code> (3N)
<code>nis_rmdir()</code> - remove directory used to support NIS+ services	<code>nis_server</code> (3N)
<code>nis_server()</code> - NIS+ functions, misc	<code>nis_server</code> (3N)
<code>nis_servstate()</code> - set and read the various state variables of the NIS+ servers	<code>nis_server</code> (3N)
<code>nis_sperrno()</code> - display NIS+ error messages	<code>nis_error</code> (3N)
<code>nis_sperror()</code> - display NIS+ error messages	<code>nis_error</code> (3N)
<code>nis_sperror_r()</code> - display NIS+ error messages	<code>nis_error</code> (3N)
<code>nis_stats()</code> - retrieve statistics about how the NIS+ server is operating	<code>nis_server</code> (3N)
<code>nis_subr()</code> - NIS+ subroutines	<code>nis_subr</code> (3N)
<code>nis_tables()</code> - NIS+ table functions	<code>nis_tables</code> (3N)
<code>nis_verifygroup()</code> - NIS+ group manipulation functions	<code>nis_groups</code> (3N)
<code>nl</code> - line numbering filter	<code>nl</code> (1)
<code>nl()</code> - enable/disable newline translation	<code>nl</code> (3X)
<code>nl_io</code> - Native Language I/O (NLIO) Subsystem	<code>nl_io</code> (5)
NLIO Subsystem, Native Language I/O	<code>nl_io</code> (5)
<code>nlist</code> - nlist structure format	<code>nlist</code> (4)
<code>nlist</code> structure format	<code>nlist</code> (4)
<code>nlist()</code> - get entries from name list	<code>nlist</code> (3C)
<code>nlist64()</code> - get entries from name list	<code>nlist</code> (3C)
<code>nljust</code> - justify lines left or right for NLS printing	<code>nljust</code> (1)
NLS (Native Language Support)	<code>glossary</code> (9)
NLS information, get locale-specific	<code>locale</code> (1)
NLS message catalog, open or close for reading	<code>catopen</code> (3C)
NLS, description of supported languages	<code>lang</code> (5)
NLS, get an NLS program message	<code>catgets</code> (3C)
NLS, information about native languages	<code>nl_langinfo</code> (3C)
NLS, language information constants	<code>langinfo</code> (5)
NLS, open or close message catalog for reading	<code>catopen</code> (3C)
NLS, query numeric formatting conventions of current locale	<code>localeconv</code> (3C)
NLS: justify lines left or right for NLS printing	<code>nljust</code> (1)
<code>NLSPATH</code>	<code>glossary</code> (9)
<code>nl_langinfo()</code> - obtain NLS string form of local language variable	<code>nl_langinfo</code> (3C)
<code>nm</code> - print name list of common object file	<code>nm</code> (1)
<code>nocbreak()</code> - input mode control functions	<code>cbreak</code> (3X)
node from a binary search tree, delete a	<code>tsearch</code> (3C)
node name	<code>glossary</code> (9)
node name, display/set	<code>uname</code> (1)
node name, get/set	<code>uname</code> (2)
node, transfer NIS database from NIS server to local	<code>ypxfr</code> (1M)
<code>nodelay()</code> - enable or disable block during read	<code>nodelay</code> (3X)
nodes on FDDI ring, list characteristics of	<code>fddinet</code> (1M)
<code>noecho()</code> - enable/disable terminal echo	<code>echo</code> (3X)
<code>nohup</code> - ignore hangups during command execution	<code>csh</code> (1)

Index

All Volumes

Description	Entry Name(Section)
nohup - run a command immune to hangups	nohup(1)
non-ASCII characters in a file, make visible or invisible	vis(1)
non-local goto, save/restore stack environment for	setjmp(3C)
nondefault priority, run a command at	nice(1)
nonl () - enable/disable newline translation	nl(3X)
nonspacing characters	glossary(9)
nop (do nothing) and return zero or non-zero exit status	true(1)
noqiflush () - enable/disable queue flushing	noqiflush(3X)
noraw () - input mode control functions	cbreak(3X)
normalized value, floating-point test for	isnormal(3M)
notify - notify user of change in job status	csh(1)
notify users of new mail in mailboxes	newmail(1)
notify you when it is time to leave	leave(1)
notimeout () - control blocking on input	notimeout(3X)
nroff, preprocess tables for	tbl(1)
nroff - format text	nroff(1)
nroff input, eliminate .so's from	soelim(1)
nroff/troff files, check	checknr(1)
nroff/troff, tbl, and neqn constructs, remove	deroff(1)
nslookup - query name servers interactively	nslookup(1)
nsquery () - query the Name Service Switch backend libraries	nsquery(1)
nsswitch.conf - configuration file for the name-service switch	nsswitch.conf(4)
ntohl () - convert values between host and network byte order	byteorder(3N)
ntohs () - convert values between host and network byte order	byteorder(3N)
NTP (Network Time Protocol), set time and date	ntptime(1M)
NTP daemon	xntpd(1M)
NTP, query program	ntp(1M)
ntptime - set time and date via NTP	ntptime(1M)
ntpq - Network Time Protocol query program	ntpq(1M)
null - null file	null(7)
nulladm - create empty file owned by adm with mode 664	acctsh(1M)
number of columns on terminal screen	COLS(3X)
number of free disk blocks, report (Berkeley version)	bdf(1M)
number of free DOS disk clusters, report	dosdf(1)
number of lines on terminal screen	LINES(3X)
number of processors installed in the system, determine	pthread_processor_bind_np(3T)
number to string or string array element, convert floating-point	ecvt(3C)
number to string, convert long double floating-point	ldcvt(3C)
number, convert string to double-precision	strtod(3C)
number, convert string to long double-precision	strtold(3C)
number, convert wide character string to double-precision	wcstod(3C)
numbering filter, line	nl(1)
numbers, generate uniformly distributed pseudo-random	drand48(3C)
numbers, inode generation, install random	fsrand(1M)
numbers, magic, for HP-UX implementations	magic(4)
numeric formatting conventions of current locale, query	localeconv(3C)
object code debugger	adb(1)
object code file, print symbol table (name list) for	nm(1)
object code files in a library, find optimum sequence for	lorder(1)
object file access library	elf(3E)
object file, ELF, finish using	elf_end(3E)
object file, link editor and assembler	a.out(4)
object file, strip symbol and line number information from	strip(1)
object files and libraries, generate single executable file from	ld(1)
object files in binary directories, install	cpset(1M)
object files, print section sizes and allocation space of	size(1)
object files: dump information contained in	elfdump(1)
object formats, NIS+	nis_objects(3N)
object into virtual memory, map	mmap(2)
object or binary file, find the printable strings in an	strings(1)
object, update the public keys in a NIS+ directory object	nisupdkeys(1M)

Description	Entry Name(Section)
object-code file, execute	exec(2)
objects, attributes, and storage formats for SD	sd(4)
objects: symbolically link	nisln(1)
obsolete library routines for RPC	rpc_soc(3N)
obtain the thread ID for the calling thread	pthread_self(3T)
ocd - outbound connection daemon used by DDFA software	ocd(1M)
ocdebug - outbound connection daemon debug utility used by DDFA software	ocdebug(1M)
octal equivalents: ASCII character set	ascii(5)
octal file dump	od(1)
od - octal file dump	od(1)
offset for an object file, get base	elf_getbase(3E)
on - execute command on a remote host	on(1)
on-line manuals, fix pages for faster viewing with <i>man(1)</i>	fixman(1M)
onintr - specify shell's treatment of interrupts	csh(1)
open a directory and associated directory stream for access	directory(3C)
open a message queue	mp_open(2)
open a shared object	dlopen(3C)
open crash dump for reading	cr_open(3)
open file description	glossary(9)
open file descriptor to a specific slot, duplicate an	dup2(2)
open file descriptor, duplicate an	dup(2)
open file	glossary(9)
open files, file control options for	fcntl(5)
open or close NLS message catalog for reading	catopen(3C)
open or close pipe I/O to or from a process	popen(3S)
open or re-open a stream file; convert file to stream	fopen(3S)
Open Shortest Path First (OSPF)	ospf_monitor(1M)
open() - open file for reading or writing	open(2)
open, access, or close a directory	directory(3C)
open-file control	fcntl(2)
open/create a named semaphore	sem_open(2)
open64() - file system API to support large files	creat64(2)
opendir() - open a directory and associated directory stream for access	directory(3C)
openlog() - initialize system log file	syslog(3C)
operating system information, display	uname(1)
operating system information, get	uname(2)
operating system name, display	uname(1)
operating system name, get	uname(2)
operating system, introduction	Introduction(9)
operating system, load (reboot)	boot(1M)
operating system, save a crash dump of	savecrash(1M)
operations on a stream file, get or reposition pointer for I/O	fseek(3S)
operations, clock	clocks(2)
operations, message control	msgctl(2)
operations, real-time scheduling	rtsched(2)
operations, semaphore control	semctl(2)
operations, semaphore	semop(2)
operations, set time limit for I/O	io_timeout_ctl(3I)
operations, shared memory control	shmctl(2)
operations, timer	timers(2)
optarg(), optind(), opterr() - get option letter from argument vector	getopt(3C)
optimization - extract error messages from C source into a file	mkstr(1)
optimize an existing HFS file system	tunefs(1M)
optimum sequence for object code files in a library, find	lorder(1)
option letter from argument vector, get	getopt(3C)
options for a non-serial printer, set printing	slp(1)
options for a terminal port, set	stty(1)
options for transport endpoint (X/OPEN TLI-XTI)	t_optmgmt(3)
options on sockets, get or set	getsockopt(2)
options, file control for open files	fcntl(5)
options, parse command	getopt(1)

Index

All Volumes

Description	Entry Name(Section)
options, parse suboptions from a string	getsubopt(3C)
options, parse utility (command)	getopts(1)
opx25 - execute HALGOL programs	opx25(1M)
order of data, convert string	strord(3C)
orderable HP-UX manuals, list of	manuals(5)
ordering relation for files in an object code library, find	lorder(1)
ordinary file	glossary(9)
ordinary file, make	mknod(2)
organization, description of HP-UX header file	stdsyms(5)
original, applying a diff file to	patch(1)
orphan process	glossary(9)
orphaned process group	glossary(9)
osdd - print or check documents formatted with the mm macros	mm(1)
OSPF gateways, monitor	ospf_monitor(1M)
ospf_monitor - monitor OSPF gateways	ospf_monitor(1M)
OTHER scheduling policy	rtsched(2)
out-bound terminal line connection, establish an	dial(3C)
outbound connection daemon debug utility used by DDFA software	ocdebug(1M)
outbound connection daemon used by DDFA software	ocd(1M)
outgoing network connections, list owners of	owners(1M)
output (format and print) files	pr(1)
output attributes to terminal	vidattr(3X)
output commands to the terminal	putp(3X)
output cursor movement commands to the terminal	mvcur(3X)
output device, finite-width, fold long lines for	fold(1)
output first few lines in a file	head(1)
output format, link editor and assembler	a.out(4)
output of a varargs argument list, print formatted	vprintf(3S)
output, formatted, print in window	mvprintw(3X)
output, formatted, print to standard output, file, or string	printf(3S)
output, standard, send copy of to specified file	tee(1)
output, terminal, control functions	clearok(3X)
output/input, buffered, standard stream file package	stdio(3S)
over-temperature environment, handle	envd(1M)
overflow monitor daemon, audit-	audomon(1M)
overlapped windows, copy	overlay(3X)
overlay() - copy overlapped windows	overlay(3X)
overview of various system shells	sh(1)
overwrite file with an existing file	cp(1)
overwrite file with an existing file	mv(1)
overwrite() - copy overlapped windows	overlay(3X)
owner and group of a file, change	chown(2)
owner and/or group, change in access control list (ACL)	chownacl(3C)
owner	glossary(9)
owner of file, change	chown(1)
owner, change owner of an NIS+ object	nischown(1)
owners - lists owners of outgoing network connections	owners(1M)
ownership, summarize file system	quot(1M)
ownership, summarize file system	quot_vxfs(1M)
pack - compress (pack) files using Huffman code (see compact(1))	pack(1)
packed decimal library, HP 3000-mode	hppac(3X)
packet filter specification file format, ppp	ppp.Filter(4)
Packet Mode module for STREAMS pty	pkt(7)
packet routing, system support for local network	routing(7)
packets, echo	ping(1M)
packets, ECHO_REQUEST	ping(1M)
packets, spray	spray(1M)
pad management functions	newpad(3X)
pad, enhanced, management function	subpad(3X)
pad, refresh immediately after writing a character rendition	pechochar(3X)
page - file perusal filter for crt viewing	more(1)

Description	Entry Name(Section)
page number, physical, validate whether dumped	cr_isaddr(3)
page size, get the current	getpagesize(2)
page, print out a manual; find manual information by keywords	man(1)
paging behavior, advise system of process's expected	madvise(2)
paging space information, system	swapinfo(1M)
paging, add swap device for interleaved	swapon(2)
paging, enable device or file system	swapon(1M)
pair_content() - color manipulation functions	can_change_color(3X)
PAM account validation procedures, perform	pam_acct_mgmt(3)
pam - Pluggable Authentication Module	pam(3)
PAM routine to retrieve user name	pam_get_user(3)
PAM routines to maintain module specific state	pam_set_data(3)
PAM	see Pluggable Authentication Modules
PAM Service Module APIs	pam_sm(3)
PAM session creation and termination operations, perform	pam_open_session(3)
PAM user policy definition service module	pam_updbe(5)
PAM, authentication information routines for PAM	pam_set_item(3)
PAM, authentication transaction routines for PAM	pam_start(3)
PAM, get error message string	pam_strerror(3)
PAM, perform authentication within the PAM framework	pam_authenticate(3)
PAM, perform password related functions within the PAM framework	pam_chauthtok(3)
PAM, service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
PAM, service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
PAM, service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
PAM, service provider implementation for pam_open_session() and pam_close_session()	pam_sm_open_session(3)
PAM, service provider implementation for pam_setcred()	pam_sm_setcred(3)
PAM, user configuration files for pluggable authentication modules	pam_user.conf(4)
pam.conf - configuration file for pluggable authentication module	pam.conf(4)
pam_acct_mgmt() - perform PAM account validation procedures	pam_acct_mgmt(3)
pam_authenticate() - perform authentication within the PAM framework	pam_authenticate(3)
pam_chauthtok() - perform password related functions within the PAM framework	pam_chauthtok(3)
pam_close_session() - perform PAM session creation and termination operations	pam_open_session(3)
pam_end() - authentication transaction routines for PAM	pam_start(3)
pam_get_data() - PAM routines to maintain module specific state	pam_set_data(3)
pam_get_item() - authentication information routines for PAM	pam_set_item(3)
pam_get_user() - PAM routine to retrieve user name	pam_get_user(3)
pam_open_session() - perform PAM session creation and termination operations	pam_open_session(3)
pam_setcred() - modify and delete user credentials for an authentication service	pam_setcred(3)
pam_set_data() - PAM routines to maintain module specific state	pam_set_data(3)
pam_set_item() - authentication information routines for PAM	pam_set_item(3)
pam_sm() - PAM Service Module APIs	pam_sm(3)
pam_sm_acct_mgmt() - Service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
pam_sm_authenticate() - service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
pam_sm_chauthtok() - service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
pam_sm_close_session() - service provider implementation for pam_close_session()	pam_sm_open_session(3)
pam_sm_open_session() - service provider implementation for pam_open_session()	pam_sm_open_session(3)
pam_sm_setcred() - service provider implementation for pam_setcred()	pam_sm_setcred(3)
pam_start() - authentication transaction routines for PAM	pam_start(3)
pam_strerror() - get PAM error message string	pam_strerror(3)
pam_unix - authentication, account, session and password management PAM modules for UNIX ..	pam_unix(5)
pam_updbe - user policy definition service module, PAM	pam_updbe(5)
pam_user.conf - user configuration file for pluggable authentication modules	pam_user.conf(4)
parallel poll on HP-IB, control response to	hpib_card_ppoll_resp(3I)
parallel poll value occurs, wait until a particular	hpib_wait_on_ppoll(3I)
parameter, system, query, set, or reset	kmtune(1M)
parameters, configure network interface	ifconfig(1M)
parameters, display system	sysdef(1M)

Index

All Volumes

Description	Entry Name(Section)
parent directory	glossary(9)
parent process	fork(2)
parent process	glossary(9)
parent process ID	glossary(9)
parent process ID, get	getpid(2)
parents, synchronise a window with	syncok(3X)
parity consistency of HP SCSI disk array LUNs	pscan(1M)
parity information, repair disk array LUN	rpr(1M)
parity information, scan for consistency	scn(1M)
parse command options	getopt(1)
parse suboptions from a string	getsubopt(3C)
parse utility (command) options	getopts(1)
parser used by DDFA software, dedicated ports	dpp(1M)
particular Network Information Service server, bind to a	yset(1M)
particular parallel poll value occurs, wait until a	hpib_wait_on_ppoll(3I)
partition DDS tape cartridge	mediainit(1)
partitions to LVM logical volumes, prepare root file system for migration from	lvmigrate(1M)
Pascal and C execution startup routines	crt0(3)
pass-through device driver, SCSI	scsi_ctl(7)
pass-through device driver, SCSI	scsi_pt(7)
passwd - change login password and associated attributes	passwd(1)
passwd database, daemon for modifying Network Information Service	yppasswdd(1M)
passwd file	login(1)
passwd file, change default login shell	chsh(1)
passwd - password file, <pwd.h> format	passwd(4)
password and group hashing and caching daemon	pwgrd(1M)
password database entry, manipulate protected	getprpwent(3)
password encryption function	crypt(3C)
password expiration	passwd(1)
password file entry, write	putpwent(3C)
password file, <pwd.h> format	passwd(4)
password file, check	pwck(1M)
password file, edit using vi editor	vipw(1M)
password file, get entry from	getpwent(3C)
password file, grp.h for user group access and identification	group(4)
password file, information used by finger , change	chfn(1)
password file, protected password database	prpwd(4)
password generation	passwd(1)
password	glossary(9)
password in Network Information Service, update user	yppasswd(3N)
password in Network Information System (NIS), change login	yppasswd(1)
password information, get (pwget)	pwget(1)
password related functions within the PAM framework, perform	pam_chauthtok(3)
password update daemon, NIS+	rpc.nispasswdd(1M)
password, authentication, account, and session management PAM modules for UNIX	pam_unix(5)
password, change login	passwd(1)
password, read from terminal while suppressing echo	getpass(3C)
password: changing NIS+ information	nispasswd(1)
passwd and group hashing and caching statistics	pwgr_stat(1M)
paste - merge corresponding lines of several files or subsequent lines of one file	paste(1)
patch - program for applying a diff file to an original	patch(1)
patch up damaged file system (generic)	fsdb(1M)
patch up damaged HFS file system	fsdb_hfs(1M)
path and route between hosts, compute shortest	pathalias(1)
PATH environment variable	login(1)
path name component	basename(3C)
path name corresponding to i-number, list	ff_hfs(1M)
path name	glossary(9)
path name of parent directory	basename(3C)
path name resolution	glossary(9)
path name variables, get configurable	pathconf(2)

Description	Entry Name(Section)
path names from i-numbers, generate	ncheck(1M)
path names, check	pathchk(1)
path names, extract portions of	basename(1)
path prefix	glossary(9)
path, map device ID to file	devnm(3)
path-name of current working directory, get	getcwd(3C)
path: create, remove directories in a path	mkdirp(3G)
pathalias database, access and manage the	uupath(1)
pathalias - electronic address router	pathalias(1)
pathchk - check path names	pathchk(1)
pathconf() - get configurable path name variables	pathconf(2)
pathfind() - search for named file in named directories	pathfind(3G)
pathname of current working directory, get	getwd(3C)
pathname, resolve	realpath(3X)
pathnames from inode numbers for VxFS file system, generate	ncheck_vxfs(1M)
paths and aliases, locate a program file including	which(1)
pattern matching and regular expression notation definitions	regex(5)
pattern scanning and processing language, text	awk(1)
patterns, match filename	fnmatch(3C)
pause execution for a time interval	sleep(1)
pause() - suspend process until signal	pause(2)
pauses jobs, physical printers, servers, or queues	pdpause(1)
pax - portable archive exchange	pax(1)
PC-NFS authentication and print request server	pcnfsd(1M)
pcat - expand (unpack) and cat Huffman coded file (see <i>compact</i> (1))	pack(1)
pcf - port configuration file, used by DDFA software	pcf(4)
pckt - Packet Mode module for STREAMS pty	pckt(7)
pclose() - terminate pipe I/O to or from a process	popen(3S)
pcnfsd - PC-NFS authentication and print request server	pcnfsd(1M)
pcserver - Basic Serial and HP AdvanceLink server	pcserver(1M)
pd - processor-dependent code (firmware)	pd(1M)
pdclean - removes all jobs from the specified object	pdclean(1)
pdcreate - creates print objects	pdcreate(1)
pddcesetup - configure DCE for the HP Distributed Print Service	pddcesetup(1M)
pddelete - deletes print objects	pddelete(1)
pddisable - stops printers from accepting jobs and logs from logging	pddisable(1)
pdenable - enables printers to accept jobs and logs to log	pdenable(1)
pdf - Product Description File format	pdf(4)
PDF, create from an input	mkpdf(1M)
pdfdiff - compare two Product Description Files	pdfdiff(1M)
pdfdiff - differences between two Product Description Files	pdfdiff(1M)
pdfdiff - Product Description Files, compare two	pdfdiff(1M)
pdgwcfg - displays the text and description of a HPDPS message at the command line	pdgwcfg(1)
pdgwcfg.conf - configuration file for HPDPS gateway printers	pdgwcfg.conf(4)
pdl - lists selected attribute values	pdclean(1)
pdmod - modifies attributes of submitted print jobs	pdmod(1)
pdmsg - displays the text and description of a HPDPS message at the command line	pdmsg(1)
pdp11 - is processor a pdp11?	machid(1)
pdpause - pauses jobs, physical printers, servers, or queues	pdpause(1)
pdpr - submits print jobs	pdpr(1)
pdpromote - advances a job to the top of a queue	pdpromote(1)
pdq - queries and lists the status of one or more print jobs	pdq(1)
pdresubmit - resubmits previously submitted print jobs	pdresubmit(1)
pdresume - enables paused objects to resume operation	pdresume(1)
pdr - removes print jobs	pdr(1)
pdset - modifies attributes of submitted print jobs	pdset(1)
pdshutdown - stop servers	pdshutdown(1)
pdstartclient - start the HPDPS client daemon	pdstartclient(1M)
pdstartspl - create or restart an HPDPS spooler	pdstartspl(1M)
pdstartsup - create or restart an HPDPS supervisor	pdstartsup(1M)
pdstop - stop the HPDPS client daemon	pdstop(1M)

Description	Entry Name(Section)
pd_att - index of attribute manpages for HP Distributed Print Service	pd_att(5)
pd_att_document - list of attributes for a document object (HP Distributed Print Service)	pd_att_document(5)
pd_att_ivdocument - list of attributes for an initial value document object (HP Distributed Print Service)	pd_att_ivdocument(5)
pd_att_ivjob - list of attributes for an initial value job object (HP Distributed Print Service)	pd_att_ivjob(5)
pd_att_job - list of attributes for a job object (HP Distributed Print Services)	pd_att_job(5)
pd_att_log - list of attributes for a log object (HP Distributed Print Service)	pd_att_log(5)
pd_att_log_ptr - list of attributes for a logical printer object (HP Distributed Print Services)	pd_att_log_ptr(5)
pd_att_phy_ptr - list of attributes for a physical printer object (HP Distributed Print Service)	pd_att_phy_ptr(5)
pd_att_queue - list of attributes for a queue object (HP Distributed Print Service)	pd_att_queue(5)
pd_att_spooler - list of attributes for a spooler object (HP Distributed Print Service)	pd_att_spooler(5)
pd_att_supervisor - list of attributes for a supervisor object (HP Distributed Print Service)	pd_att_supervisor(5)
pechochar () - write a character rendition and immediately refresh the pad	pechochar(3X)
pecho_wchar () - write a character rendition and immediately refresh the pad	pechochar(3X)
peer, get address of connected	getpeername(2)
pending cancellation requests, process any	pthread_testcancel(3T)
pending signals, examine	sigpending(2)
per-process accounting file format	acct(4)
per-process timer, allocate a	mktimer(3C)
per-process timer, free a	rmtimer(3C)
per-process timer, get value of a	gettimer(3C)
per-process timer, relatively arm a	reltimer(3C)
per-session accounting records, convert login/logoff records to	acctcon(1M)
per-session records, convert to total accounting records	acctcon(1M)
perform authentication within the PAM framework	pam_authenticate(3)
perform Cyclical Redundancy Check on a file	sum(1)
perform I/O on kernel memory based on symbol name	kmem(7)
perform I/O with an HP-IB channel from buffers	hpiob_io(3I)
perform NFS mount to remote file system	vhe_u_mnt(1M)
perform PAM account validation procedures	pam_acct_mgmt(3)
perform PAM session creation and termination operations	pam_open_session(3)
perform password related functions within the PAM framework	pam_chauthtok(3)
perform word expansions	wordexp(3C)
performance analysis information, print LP spooler	lpana(1M)
performance data from remote kernel, get	rstat(3N)
periodic accounting summary files, create	acctsh(1M)
permission bits	glossary(9)
permissions file, check the uucp directories and	uucheck(1M)
permissions mask for file creation, set and get	umask(2)
permissions mode mask for file-creation, set access	umask(1)
permissions, change file mode access	chmod(1)
permissions, change file mode access	chmod(2)
permit or deny <i>write</i> (1) messages from other users to terminal	mesg(1)
permuted index, generate	ptx(1)
perror() , errno() , sys_errlist() , sys_nerr() - system error messages	perror(3C)
peruse file on CRT terminals	more(1)
peruse file on soft-copy terminals	pg(1)
pfmt() - display message in standard format	pfmt(3C)
pfmt() - message, display in standard format	pfmt(3C)
pfmt() - standard format, display message in	pfmt(3C)
PFS clients, directories to export to	pfs_exports(5)
PFS clients, export and unexport directories to	pfs_exportfs(1M)
PFS daemon	pfsd(1M)
PFS mount request server	pfs_mountd(1M)
pfs - PFS, portable file system	pfs(4)
PFS, portable file system	pfs(4)
pfsd - PFS daemon	pfsd(1M)

Description	Entry Name(Section)
pfs_exportfs - export and unexport directories to PFS clients	pfs_exportfs(1M)
pfs_exports - directories to export to PFS clients	pfs_exports(5)
pfs_fstab - static file system mounting table	pfs_fstab(5)
pfs_mount - mount and unmount CD-ROM file systems	pfs_mount(1M)
pfs_mountd - PFS mount request server	pfs_mountd(1M)
pg - file perusal filter for soft-copy terminals	pg(1)
phone message transcription system	answer(1)
physical device description file format, for ppp	ppp.Devices(4)
physical environment daemon, system	envd(1M)
physical extents allocated to LVM logical volume, decrease	lvreduce(1M)
physical extents, move from one LVM physical volume to other physical volumes	pvmove(1M)
physical memory address mapping	iomap(7)
physical page number, validate whether dumped	cr_isaddr(3)
physical volume (LVM) to other physical volumes, move allocated physical extents from	pvmove(1M)
physical volume for use in LVM volume group, create	pvccreate(1M)
physical volume group information file (LVM)	lvmpvg(4)
physical volume in LVM volume group, change characteristics and access path	pvchange(1M)
physical volume in LVM volume group, check or repair a	pvck(1M)
physical volume	lvm(7)
physical volumes in LVM volume group, display information about	pvddisplay(1M)
physical volumes, extend an LVM volume group by adding	vgextend(1M)
physical volumes, reduce LVM volume group by removing	vgreduce(1M)
physical volumes, scan for LVM volume groups	vgscan(1M)
PIC	glossary(9)
ping - send echo packets to a network host	ping(1M)
ping, send to NIS+ servers	nisping(1M)
pipe fitting	tee(1)
pipe	glossary(9)
pipe I/O to or from a process, open or close	popen(3S)
pipe() - create an interprocess channel	pipe(2)
pipe: STREAMS-based pipe	isastream(3C)
pipes, create named	mknod(1M)
play audio file	send_sound(1)
plock() - lock process, text, data, stack, or shared library in memory	plock(2)
plotdvr - plotter driver for lp	lpfilter(1)
plotter	see printer
pluggable authentication module	pam(3)
pluggable authentication module, configuration file	pam.conf(4)
Pluggable Authentication Modules	login(1)
pmmap_getmaps() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_getport() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_rmtcall() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_set() - obsolete library routines for RPC	rpc_soc(3N)
pmmap_unset() - obsolete library routines for RPC	rpc_soc(3N)
pnoutrefresh() - pad management functions	newpad(3X)
point to point protocol authentication file format	ppp.Auth(4)
point to point protocol daemon	cp(1)
point to point protocol physical device description file format	ppp.Devices(4)
pointer array, sort a directory	scandir(3C)
pointer for binary search tree, get data	tsearch(3C)
pointer for I/O operations on a stream file, get or reposition	fseek(3S)
pointer, file, move read/write	lseek(2)
pointer, stream, map to file descriptor	fileno(3S)
pointer, string, for ELF files, make	elf_strptr(3E)
poll on HP-IB bus, conduct a serial	hpiib_spoll(3I)
poll on HP-IB, control response to parallel	hpiib_card_ppoll_resp(3I)
poll value occurs, wait until a particular parallel	hpiib_wait_on_ppoll(3I)
poll() - monitor I/O conditions on multiple file descriptors	poll(2)
pong - send Fibre Channel Light Weight Protocol Echo Request packet	pong(1M)
popd - pop directory stack	csh(1)
popen() - initiate pipe I/O to or from a process	popen(3S)

Index

All Volumes

Description	Entry Name(Section)
populate the NIS+ tables in a NIS+ domain	nispopulate(1M)
port number, RPC, get	getrpcport(3N)
port socket, return a reserved	rcmd(3N)
port, IP, bind socket to a privileged	bindresvport(3N)
portable archive exchange	pax(1)
portable archives, library and archive maintainer for	ar(1)
portable file name character set	glossary(9)
portable file system, PFS	pfs(4)
portal - header file for future applications	portal(5)
portions of path names, extract	basename(1)
ports file used by DDFA software and Telnet port identification feature, dedicated	dp(4)
ports parser used by DDFA software, dedicated	dpp(1M)
position-independent code (PIC)	glossary(9)
positive difference function	fdim(3M)
POSIX asynchronous I/O	aio(5)
POSIX configuration values, get	getconf(1)
POSIX real-time priority, execute process with	rtsched(1)
POSIX realtime extensions	aio(5)
POSIX realtime extensions	aio_cancel(2)
POSIX realtime extensions	aio_error(2)
POSIX realtime extensions	aio_fsync(2)
POSIX realtime extensions	aio_read(2)
POSIX realtime extensions	aio_return(2)
POSIX realtime extensions	aio_suspend(2)
POSIX realtime extensions	aio_write(2)
POSIX realtime extensions	lio_listio(2)
POSIX realtime extensions	mlock(2)
POSIX realtime extensions	mlockall(2)
POSIX realtime extensions	munlock(2)
POSIX realtime extensions	munlockall(2)
POSIX semaphore, close a named semaphore	sem_close(2)
POSIX semaphore, destroy an unnamed semaphore	sem_destroy(2)
POSIX semaphore, initialize an unnamed semaphore	sem_init(2)
POSIX semaphore, lock a semaphore	sem_wait(2)
POSIX semaphore, lock a semaphore without blocking	sem_wait(2)
POSIX semaphore, open/create a named semaphore	sem_open(2)
POSIX semaphore, read	sem_getvalue(2)
POSIX semaphore, unlink a named semaphore	sem_unlink(2)
POSIX semaphore, unlock a semaphore	sem_post(2)
POSIX.1c threads	pthread(3T)
posters, make in large letters	banner(1)
pow() , powf() - power function	pow(3M)
power functions	pow(3M)
power_onoff - timed, automatic system power on, and power off	power_onoff(1M)
powf() , pow() - power function (float version)	pow(3M)
ppp authentication file format	ppp.Auth(4)
ppp dialer description file format	ppp.Dialers(4)
ppp encryption keys file format	ppp.Keys(4)
ppp neighboring systems description file format	ppp.Systems(4)
ppp packet filter specification file format	ppp.Filter(4)
ppp physical device description file format	ppp.Devices(4)
ppp.auth - ppp authentication file format	ppp.Auth(4)
ppp.Devices - ppp physical device description file format	ppp.Devices(4)
ppp.Dialers - ppp dialer description file format	ppp.Dialers(4)
ppp.Filter - ppp packet filter specification file format	ppp.Filter(4)
ppp.Keys - ppp encryption keys file format	ppp.Keys(4)
ppp.Systems - ppp neighboring systems description file format	ppp.Systems(4)
pppd - point to point protocol daemon	pppd(1)
pr - format and print files	pr(1)
praliases - print system-wide sendmail aliases	praliases(1)
prcmd - return streams to parallel remote commands	prcmd(3N)

Description	Entry Name(Section)
prctmp - print session record file created by acctcon1	acctsh(1M)
prdaily - print daily accounting report	acctsh(1M)
prealloc - preallocate disk storage	prealloc(1)
prealloc() - preallocate fast disk storage	prealloc(2)
prealloc64() - file sysmmmaptem API to support large files	creat64(2)
preallocate fast disk storage	prealloc(2)
preallocate space for a disk storage file	prealloc(1)
Precision Architecture assembler	as(1)
prefix characters to obtain text line of specified length	newform(1)
prefresh() - pad management functions	newpad(3X)
prepare an incomplete executable for faster program start-up	fastbind(1)
prepare execution profile	monitor(3C)
prepare LVM logical volume to be root, swap, or dump volume	lvlnboot(1M)
prepare root file system for migration from partitions to LVM logical volumes	lvmigrate(1M)
preprocess mathematical text for nroff	neqn(1)
preprocess tables for nroff	tbl(1)
preprocessor lines, remove	unifdef(1)
preprocessor , C language	cpp(1)
preset contents of memory area to specified byte	memory(3C)
prevent terminal use by others	lock(1)
previous get of an SCCS file, undo a	unget(1)
primary swap volume, prepare LVM logical volume to be	lvlnboot(1M)
primary swap volume, remove LVM logical volume link	lvrmboot(1M)
primes , factor - factor a number, generate large primes	factor(1)
primitive system data types	types(5)
principals , NIS+, initialize NIS+ credentials for	nisclient(1M)
print (echo) arguments	echo(1)
print a libcrash error or warning message	cr_perror(3)
print all values in a Network Information Service map	ypcat(1)
print and summarize an SCCS file	prs(1)
print any total accounting (tacct) file	acctsh(1M)
print calendar	cal(1)
print checksum and block count of a file	sum(1)
print checksum and block count of a file	sum(1)
print current SCCS file editing activity	sact(1)
print current system-clock date and time	date(1)
print files, format and	pr(1)
print first few lines in a file	head(1)
print formatted arguments	printf(1)
print formatted output in a window	vwprintw(3X)
print formatted output in a window	vw_printw(3X)
print formatted output in window	mvprintw(3X)
print formatted output of a varargs argument list	vprintf(3S)
print formatted output to standard output, file, or string	printf(3S)
print information about local LicensePower/iFOR target id	i4target(1M)
print list of current system users	who(1)
print log messages and other information about RCS files	rlog(1)
print LP request status information	lpstat(1)
print LP spooler performance analysis information	lpana(1M)
print mail traffic statistics	mailstats(1)
print name list of object code file	nm(1)
print name of current HP-UX model	model(1)
print news items	news(1)
print or check documents formatted with the mm macros	mm(1)
print or display effective current user id	whoami(1)
print out a manual entry; find manual information by keywords	man(1)
print out mail in the incoming mailbox file	prmail(1)
print out or compare terminfo descriptions	infocmp(1M)
print out the contents of the shared cache file, NIS+	nisshowcache(1M)
print out the environment	printenv(1)
prnt - output from shell	ksh(1)

Index

All Volumes

Description	Entry Name(Section)
print - output from shell	sh-posix(1)
print process accounting files	acctcom(1M)
print request server, PC-NFS	pcnfsd(1M)
print requests on an LP printer	lp(1)
print section sizes and allocation space of object files	size(1)
print session record file created by acctcon1	acctsh(1M)
print symbol table for object code file	nm(1)
print system-wide sendmail aliases	praliases(1)
print the values of selected keys in Network Information Service map	ypmatch(1)
print user and group IDs and names	id(1)
print working directory name	pwd(1)
print, copy, and concatenate files	cat(1)
printable representation of a character, generate	unctrl(3X)
printable representation of a wide character, generate	wunctrl(3X)
printable strings in an object or other binary file, find the	strings(1)
printenv - print out the environment	printenv(1)
printer capability database	terminfo(4)
printer daemon for LP requests from remote systems	rlpdaemon(1M)
printer device files, line	lp(7)
printer for use with tsm , add or remove a	tsm.lpadmin(1M)
printer spooling system	see LP
printer, LP, allow or prevent queuing requests	accept(1M)
printer, LP, print/alter/cancel requests	lp(1)
printer, set printing options for a non-serial	slp(1)
printers, LP, enable/disable	enable(1)
printf - print formatted arguments	printf(1)
printf() - print formatted output to standard output	printf(3S)
printing options for a non-serial printer, set	slp(1)
prints the mail queue	mailq(1)
printstat - check status of serial printer	lpfilter(1)
printw() - print formatted output in window	mvprintw(3X)
pricoeiling attribute, get or set	pthread_mutexattr_getprotocol(3T)
priority of a process, change	nice(2)
priority of running processes, alter	renice(1)
priority	renice(1)
priority, get or set process	getpriority(2)
priority, POSIX real-time, execute process with	rtsched(1)
priority, real-time, execute process with	rtprio(1)
priority, run a command at nondefault	nice(1)
private encryption key storage, server for	keyserv(1)
privgrp - format of privileged values	privgrp(4)
privgrp, get special attributes for group	getprivgrp(1)
privilege group, get or set special attributes	getprivgrp(2)
privileged groups	glossary(9)
privileged IP port, bind socket to a	bindresvport(3N)
privileged values, format of	privgrp(4)
privileges for group, list access	getprivgrp(1)
privileges, associate group with certain	setprivgrp(1M)
PRI_HPUX_TO_POSIX() - return POSIX process priority	rtsched(2)
PRI_POSIX_TO_HPUX() - return HP-UX process priority	rtsched(2)
PRM	see Process Resource Manager
prmail - print out mail in the incoming mailbox file	prmail(1)
process 1	glossary(9)
process 16-bit characters, tools to	nl_tools_16(3X)
process accounting files, search and print	acctcom(1M)
process accounting	see accounting
process accounting, daily accounting shell procedure	runacct(1M)
process accounting, enable or disable	acct(2)
process and child process times, get	times(2)
process any pending cancellation requests	pthread_testcancel(3T)
process C language include and conditional instructions	cpp(1)

Description	Entry Name(Section)
process environment, clear the	clearenv(3C)
process	glossary(9)
process group	glossary(9)
process group ID for job control, set	setpgid(2)
process group ID	glossary(9)
process group ID, create session and set	setsid(2)
process group ID, foreground, get	tcgetpgrp(3C)
process group ID, foreground, set	tcsetpgrp(3C)
process group ID, get	getpid(2)
process group ID, set	setpgrp(2)
process group leader	glossary(9)
process group lifetime	glossary(9)
process ID	glossary(9)
process ID, get	getpid(2)
process interval timer, set or get value of	getitimer(2)
process lifetime	glossary(9)
process mail through screen-oriented interface	elm(1)
process or group of processes, send signal	kill(2)
process priority, get or set	getpriority(2)
Process Resource Manager	acctcom(1M)
Process Resource Manager	cron(1M)
Process Resource Manager	exec(2)
Process Resource Manager	login(1)
Process Resource Manager	ps(1)
process status, report	ps(1)
process to stop or terminate, wait for child	wait(2)
process trace	ptrace(2)
process's alarm clock, set	alarm(2)
process's expected paging behavior, advise system of	madvise(2)
process, calling, set or clear auditing on	setaudproc(2)
process, change priority of a	nice(2)
process, child, wait to change state	wait3(2)
process, child, wait to change state	waitid(2)
process, create a new	fork(2)
process, execute, with POSIX real-time priority	rtsched(1)
process, execute, with real-time priority	rtprio(1)
process, force target to run serially with other processes	serialize(2)
process, force target to run serially with other processes	serialize(1)
process, get audit ID (aid()) for current	getaudid(2)
process, get audit process flag for calling	getaudproc(2)
process, lock address space in memory	mlockall(2)
process, lock in memory	plock(2)
process, lock into memory after allocating data and stack space	datalock(3C)
process, lock segment in memory	mlock(2)
process, open or close pipe I/O to or from a	popen(3S)
process, queue a signal to a	sigqueue(2)
process, self-auditing, write audit record for	audwrite(2)
process, send signal to	kill(1)
process, set audit ID (aid()) for current	setaudid(2)
process, spawn new (use fork() instead)	vfork(2)
process, suspend or resume auditing on current	audswitch(2)
process, suspend the calling process	napms(3X)
process, suspend until signal	pause(2)
process, terminate	exit(2)
process, terminate	kill(1)
process, unlock memory segment	munlock(2)
process, unlock virtual address space	munlockall(2)
process-shared attribute, get or set	pthread_condattr_getpshared(3T)
process-shared attribute, get or set	pthread_mutexattr_getpshared(3T)
process-shared attribute, get or set	pthread_rwlockattr_getpshared(3T)
processes and users, list current	who(1M)

Index

All Volumes

Description	Entry Name(Section)
processes currently using a file or file structure	fuser(1M)
processes on system, display and update information about top	top(1)
processes to complete, wait for background	wait(1)
processes, alter priority of running	renice(1)
processes, kill all active	killall(1M)
processes, Network Information Service (NIS) binder	ypserv(1M)
processes, spawn	init(1M)
processing language, text pattern scanning and	awk(1)
processing system, interactive mail message	mailx(1)
processor for C, Ratfor and other programming language macros	m4(1)
processor IDs, determined	pthread_processor_bind_np(3T)
processor initialization	pdc(1M)
processor self test	pdc(1M)
processor type, determine	sysconf(2)
processor-dependent code (firmware)	pdc(1M)
processors, bind threads to	pthread_processor_bind_np(3T)
processors, how many installed in the system	pthread_processor_bind_np(3T)
Product Description File format	pdf(4)
Product Description File, check against file system	pdfck(1M)
Product Description File, create from an input	mkpdf(1M)
product specification file format	swpackage(4)
products, verify software	swverify(1M)
prof - display monitor profile data	prof(1)
profil() - execution time profile	profil(2)
profile data, display call graph execution	gprof(1)
profile data, display monitor	prof(1)
profile file	login(1)
profile of execution, prepare	monitor(3C)
profile - shell script to set up user's environment at login	profile(4)
profile, execution time	profil(2)
program assertion, verify	assert(3X)
program file, locate, including aliases and paths	which(1)
program files that execute under given command name, find	which(1)
program for SCCS commands	sccs(1)
program	glossary(9)
program loaded module	dlget(3C)
program loaded module	dlmodinfo(3C)
program message, get an NLS	catgets(3C)
program number database, RPC	rpc(4)
program regions, first locations beyond allocated	end(3C)
program start-up, prepare for faster	fastbind(1)
program termination, register a function to be called at	atexit(2)
program's internal attributes, change	chatr(1)
program, get or set the locale of a	setlocale(3C)
programming language macro processor	m4(1)
programming values and constants, machine-dependent	values(5)
programs from a disk device; update, install, or remove boot	mkboot(1M)
programs - maintain, update, and regenerate groups of	make(1)
programs, a compiler/interpreter for modest-sized	bs(1)
programs, HALGOL, execute	opx25(1M)
programs, input editor and command history for interactive	ied(1)
prompt	glossary(9)
propagation of a Network Information Service database, force	yppush(1M)
protect terminal from use by others	lock(1)
protected password database entry, manipulate	getprpwent(3)
protections, modify memory mapping access	mprotect(2)
proto - prototype job file for at	proto(4)
protocol address (X/OPEN XTI)	t_getprotaddr(3)
protocol attribute, get or set	pthread_mutexattr_getprotocol(3T)
protocol entry, get, set, or end	getprotoent(3N)
protocol family, Internet	inet(7F)

Description	Entry Name(Section)
protocol module, remote magnetic tape dump and restore	rmt(1M)
protocol name database	protocols(4)
protocol server, file transfer	ftpd(1M)
protocol server, for TCP/IP IDENT	identd(1M)
protocol server, TELNET	telnetd(1M)
protocol server, trivial file transfer	tftpd(1M)
protocol, address resolution	arp(7P)
protocol, Internet user datagram	UDP(7P)
Protocol, IP Internet	IP(7P)
protocol, local communication domain	UNIX(7P)
Protocol, TCP Internet Transmission Control	TCP(7P)
protocol, user interface to the TELNET	telnet(1)
protocol-specific service information (X/OPEN TLI-XTI)	t_getinfo(3)
protocols - protocol name database	protocols(4)
prototype job file for at	proto(4)
provide semaphores and record locking on files	lockf(2)
provide sequential archive member access for ELF files	elf_next(3E)
prpwd - protected password database	prpwd(4)
prs - print and summarize an SCCS file	prs(1)
prtacct - print any total accounting (acct) file	acctsh(1M)
ps - report process status	ps(1)
PS/2 keyboard and mouse device driver	ps2(7)
ps2 - PS/2 keyboard and mouse device driver	ps2(7)
ps2kbd - PS/2 keyboard device file	ps2(7)
ps2mouse - PS/2 mouse device file	ps2(7)
pscan - scan HP SCSI disk array LUNs for parity consistency	pscan(1M)
pseudo-random numbers, generate uniformly distributed	drand48(3C)
pseudo-terminal driver	pty(7)
pseudo-terminal master driver, STREAMS	ptm(7)
pseudo-terminal slave driver, for STREAMS	pts(7)
pseudo-terminal slave driver, for STREAMS	tels(7)
pseudo-terminal, get name of user's terminal or	tty(1)
pseudo-terminal, Packet Mode module for STREAMS pty	pkct(7)
pseudo-terminal, STREAMS Emulation module	ptem(7)
pseudorandom number generation functions	random(3M)
PSF format	swpackage(4)
pstat() - get system information	pstat(2)
pstat() - set system information	pstat(2)
pstat_getdisk() - get disk information	pstat(2)
pstat_getdynamic() - get dynamic system information	pstat(2)
pstat_getfile() - get open file information	pstat(2)
pstat_getipc() - get IPC information	pstat(2)
pstat_getlv() - get logical volume information	pstat(2)
pstat_getlwp() - get lightweight process or thread information	pstat(2)
pstat_getmsg() - get message queue information	pstat(2)
pstat_getproc() - get information about a process	pstat(2)
pstat_getprocessor() - get information about a processor1	pstat(2)
pstat_getprocv() - get process address space information	pstat(2)
pstat_getsem() - get semaphore set information	pstat(2)
pstat_getshm() - get shared memory segment information	pstat(2)
pstat_getstable() - get information contained in system's stable storage area	pstat(2)
pstat_getstatic() - get static system information	pstat(2)
pstat_getswap() - get swap area information	pstat(2)
pstat_getvminfo() - get virtual memory information	pstat(2)
ptem - STREAMS pty (pseudo-terminal) Emulation module	ptem(7)
pthread register fork handler	pthread_atfork(3T)
pthread() - introduction to POSIX.1c threads	pthread(3T)
pthread_atfork() - register fork handler	pthread_atfork(3T)
pthread_attr_destroy() - destroy a thread attribute object	pthread_attr_destroy(3T)
pthread_attr_getdetachstate() - get the detachstate attribute	pthread_attr_getdetachstate(3T)
pthread_attr_getguardsize() - get the guardsize attribute	pthread_attr_getdetachstate(3T)

Description	Entry Name(Section)
<code>pthread_attr_getinheritsched()</code> - get the inheritsched attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getprocessor_np()</code> - get the processor and binding_type attributes	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getschedparam()</code> - get the schedparam attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getschedpolicy()</code> - get the schedpolicy attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getscope()</code> - get the contention scope attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getstackaddr()</code> - get the stackaddr attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_getstacksize()</code> - get the stacksize attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_init()</code> - initialize a thread attribute object	<code>pthread_attr_init(3T)</code>
<code>pthread_attr_setdetachstate()</code> - set the detachstate attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setguardsize()</code> - set the guardsize attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setinheritsched()</code> - set the inheritsched attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setprocessor_np()</code> - set the processor and binding_type attributes	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setschedparam()</code> - set the schedparam attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setschedpolicy()</code> - set the schedpolicy attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setscope()</code> - set the contention scope attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setstackaddr()</code> - set the stackaddr attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_attr_setstacksize()</code> - set the stacksize attribute	<code>pthread_attr_getdetachstate(3T)</code>
<code>pthread_cancel()</code> - cancel execution of a thread	<code>pthread_cancel(3T)</code>
<code>pthread_cleanup_pop()</code> - remove a thread cancellation cleanup handler	<code>pthread_cleanup_pop(3T)</code>
<code>pthread_cleanup_push()</code> - register a thread cancellation cleanup handler	<code>pthread_cleanup_pop(3T)</code>
<code>pthread_condattr_destroy()</code> - destroy a thread condition variable attributes object	<code>pthread_condattr_init(3T)</code>
<code>pthread_condattr_getpshared()</code> - get the thread process-shared attribute	<code>pthread_condattr_getpshared(3T)</code>
<code>pthread_condattr_init()</code> - initialize a thread condition variable attributes object	<code>pthread_condattr_init(3T)</code>
<code>pthread_condattr_setpshared()</code> - set the thread process-shared attribute	<code>pthread_condattr_getpshared(3T)</code>
<code>pthread_cond_broadcast()</code> - unblock all threads waiting on a condition variable	<code>pthread_cond_signal(3T)</code>
<code>pthread_cond_destroy()</code> - destroy a thread condition variable	<code>pthread_cond_init(3T)</code>
<code>pthread_cond_init()</code> - initialize a thread condition variable	<code>pthread_cond_init(3T)</code>
<code>pthread_cond_signal()</code> - unblock one thread waiting on a condition variable	<code>pthread_cond_signal(3T)</code>
<code>pthread_cond_timedwait()</code> - timedwait on a thread condition variable	<code>pthread_cond_wait(3T)</code>
<code>pthread_cond_wait()</code> - wait on a thread condition variable	<code>pthread_cond_wait(3T)</code>
<code>pthread_continue()</code> - continue execution of a thread	<code>pthread_resume_np(3T)</code>
<code>pthread_create()</code> - create a new thread of execution	<code>pthread_create(3T)</code>
<code>pthread_default_stacksize_np()</code> - change the default stacksize	<code>pthread_default_stacksize_np(3T)</code>
<code>pthread_detach()</code> - mark a thread as detached to reclaim its resources when terminate	<code>pthread_detach(3T)</code>
<code>pthread_equal()</code> - compare two thread identifiers	<code>pthread_equal(3T)</code>
<code>pthread_exit()</code> - cause the calling thread to terminate	<code>pthread_exit(3T)</code>
<code>pthread_getconcurrency()</code> - get concurrency level of unbound threads	<code>pthread_getconcurrency(3T)</code>
<code>pthread_getschedparam()</code> - get the scheduling policy and associated parameters	<code>pthread_getschedparam(3T)</code>
<code>pthread_getspecific()</code> - get the thread-specific data associated with a key	<code>pthread_getspecific(3T)</code>
<code>pthread_join()</code> - wait for the termination of a specified thread	<code>pthread_join(3T)</code>
<code>pthread_key_create()</code> - create a thread-specific data key	<code>pthread_key_create(3T)</code>
<code>pthread_key_destroy()</code> - destroy a thread-specific data key	<code>pthread_key_create(3T)</code>
<code>pthread_kill()</code> - send a signal to a thread	<code>pthread_kill(3T)</code>
<code>pthread_mutexattr_destroy()</code> - destroy a mutex attribute object	<code>pthread_mutexattr_init(3T)</code>
<code>pthread_mutexattr_getprioceiling()</code> - get the prioceiling attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_getprotocol()</code> - get the protocol attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_getpshared()</code> - get the process-shared attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_getspin_np()</code> - get the spin attribute	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutexattr_gettype()</code> - get the type attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_init()</code> - initialize a mutex attribute object	<code>pthread_mutexattr_init(3T)</code>
<code>pthread_mutexattr_setprioceiling()</code> - set the prioceiling attribute	<code>pthread_mutexattr_getprotocol(3T)</code>
<code>pthread_mutexattr_setprotocol()</code> - set the protocol attribute	<code>pthread_mutexattr_getprotocol(3T)</code>

Description	Entry Name(Section)
<code>pthread_mutexattr_setpshared()</code> - set the process-shared attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutexattr_setspin_np()</code> - set the spin attribute	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutexattr_settype()</code> - set the type attribute	<code>pthread_mutexattr_getpshared(3T)</code>
<code>pthread_mutex_destroy()</code> - destroy a mutex	<code>pthread_mutex_init(3T)</code>
<code>pthread_mutex_getprioceiling()</code> - get the prioceiling of a mutex	<code>pthread_mutex_getprioceiling(3T)</code>
<code>pthread_mutex_getyieldfreq_np()</code> - get the yield frequency attribute	
	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutex_init()</code> - initialize a mutex	<code>pthread_mutex_init(3T)</code>
<code>pthread_mutex_lock()</code> - lock a mutex	<code>pthread_mutex_lock(3T)</code>
<code>pthread_mutex_setprioceiling()</code> - set the prioceiling of a mutex	<code>pthread_mutex_getprioceiling(3T)</code>
<code>pthread_mutex_setyieldfreq_np()</code> - set the yield frequency attribute	
	<code>pthread_mutexattr_getspin_np(3T)</code>
<code>pthread_mutex_trylock()</code> - try to lock a mutex	<code>pthread_mutex_lock(3T)</code>
<code>pthread_mutex_unlock()</code> - unlock a mutex	<code>pthread_mutex_unlock(3T)</code>
<code>pthread_num_processor_np()</code> - return how many processors are installed in the system	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_once()</code> - call an initialization routine only once	<code>pthread_once(3T)</code>
<code>pthread_processor_bind_np()</code> - bind threads to processors	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_processor_id_np()</code> - determine processor IDs	<code>pthread_processor_bind_np(3T)</code>
<code>pthread_resume_np()</code> - resume execution of a thread	<code>pthread_resume_np(3T)</code>
<code>pthread_rwlockattr_destroy()</code> - destroy a read-write lock attribute object	<code>pthread_rwlockattr_init(3T)</code>
<code>pthread_rwlockattr_getpshared()</code> - get the process-shared attribute	
	<code>pthread_rwlockattr_getpshared(3T)</code>
<code>pthread_rwlockattr_init()</code> - initialize a read-write lock attribute object	<code>pthread_rwlockattr_init(3T)</code>
<code>pthread_rwlockattr_setpshared()</code> - set the process-shared attribute	
	<code>pthread_rwlockattr_getpshared(3T)</code>
<code>pthread_rwlock_destroy()</code> - destroy a read-write lock	<code>pthread_rwlock_init(3T)</code>
<code>pthread_rwlock_init()</code> - initialize a read-write lock	<code>pthread_rwlock_init(3T)</code>
<code>pthread_rwlock_rdlock()</code> - lock a read-write lock for reading	<code>pthread_rwlock_rdlock(3T)</code>
<code>pthread_rwlock_tryrdlock()</code> - attempt to lock a read-write lock for reading	<code>pthread_rwlock_rdlock(3T)</code>
<code>pthread_rwlock_trywrlock()</code> - attempt to lock a read-write lock for writing	<code>pthread_rwlock_wrlock(3T)</code>
<code>pthread_rwlock_unlock()</code> - unlock a read-write lock	<code>pthread_rwlock_unlock(3T)</code>
<code>pthread_rwlock_wrlock()</code> - lock a read-write lock for writing	<code>pthread_rwlock_wrlock(3T)</code>
<code>pthread_self()</code> - obtain the thread ID for the calling thread	<code>pthread_self(3T)</code>
<code>pthread_setcancelstate()</code> - set and retrieve the current thread's cancelability state	<code>pthread_setcancelstate(3T)</code>
	<code>pthread_setcancelstate(3T)</code>
<code>pthread_setcanceltype()</code> - set and retrieve the current thread's cancelability type	
	<code>pthread_setcancelstate(3T)</code>
<code>pthread_setconcurrency()</code> - set concurrency level of unbound threads	<code>pthread_getconcurrency(3T)</code>
<code>pthread_setschedparam()</code> - set the scheduling policy and associated parameters	<code>pthread_getschedparam(3T)</code>
<code>pthread_setspecific()</code> - set the thread-specific data associated with a key	<code>pthread_getspecific(3T)</code>
<code>pthread_sigmask()</code> - examine and change the signal mask of the calling thread	<code>pthread_sigmask(3T)</code>
<code>pthread_suspend()</code> - suspend execution of a thread	<code>pthread_resume_np(3T)</code>
<code>pthread_testcancel()</code> - process any pending cancellation requests	<code>pthread_testcancel(3T)</code>
<code>ptm</code> - STREAMS master pty (pseudo-terminal) driver	<code>ptm(7)</code>
<code>ptrace()</code> - process trace	<code>ptrace(2)</code>
<code>pts</code> - STREAMS slave pty (pseudo-terminal) driver	<code>pts(7)</code>
<code>ptsname()</code> - get the name of a slave pty	<code>ptsname(3C)</code>
<code>ptsname_r()</code> - get the name of a slave pty	<code>ptsname(3C)</code>
<code>ptx</code> - generate permuted index	<code>ptx(1)</code>
<code>pty</code> - get the name of the user's pseudo-terminal	<code>tty(1)</code>
<code>pty</code> master driver, STREAMS	<code>ptm(7)</code>
<code>pty</code> - pseudo-terminal driver	<code>pty(7)</code>
<code>pty</code> , get the name of a slave	<code>ptsname(3C)</code>
public key, database	<code>publickey(4)</code>
public keys, update, in a NIS+ directory object	<code>nisupdkeys(1M)</code>
public or secret key, retrieve	<code>getpublickey(3M)</code>
public UNIX system to UNIX system file copy	<code>uuto(1)</code>
publickey database file, creating new key in	<code>newkey(1M)</code>
publickey database file, updates to	<code>udpublickey(1M)</code>

Index

All Volumes

Description	Entry Name(Section)
publickey - database for public keys	publickey(4)
publickey() - retrieve public or secret key	getpublickey(3M)
push a character onto the input queue	ungetch(3X)
push character back into input stream	ungetc(3S)
push wide character back into input stream	ungetwc(3C)
pushAgent - install Software Distributor agent on remote systems	pushAgent(1M)
pushd - push directory stack	csh(1)
pushed STREAMS modules, manage system database	autopush(1M)
put a string on a stream	puts(3S)
put character or word on a stream	putc(3S)
put wide character on a stream	putwc(3C)
put word or character on a stream	putc(3S)
putc(), fputc() - put character on a stream	putc(3S)
putchar() - put character on stream standard output	putc(3S)
putdvagname() - add or rewrite device assignment database entry	getdvagent(3)
putenv() - change or add value to environment	putenv(3C)
putmsg() - send a message on a stream	putmsg(2)
putp() - output commands to the terminal	putp(3X)
putpmsg() - send a message on a stream in different priority bands	putmsg(2)
putprdfname() - manipulate system default database entry	getprdfent(3)
putprpwnam() - manipulate protected password database entry	getprpwent(3)
putprtcnam() - manipulate terminal control database entry	getprtcent(3)
putpwent() - write password file entry	putpwent(3C)
puts() - write null-terminated string to stream stdout()	puts(3S)
pututline() - update or create entry in a utmp() file	getut(3C)
pututline_r() - update or create entry in a utmp() file	getut(3C)
pututxline() - update or create entry in a utmpx() file	getutx(3C)
putw() - put word (integer) on a stream	putc(3S)
putwc(), fputwc() - put wide character on a stream	putwc(3C)
putwchar() - put wide character on stream standard output	putwc(3C)
putwin() - dump window to and reload window from a file	getwin(3X)
putws(), fputws() - write null-terminated wide string to a named stream file	putws(3C)
pvchange - change characteristics and access path of physical volume in LVM volume group	pvchange(1M)
pvck - check or repair a physical volume in LVM volume group	pvck(1M)
pvccreate - create physical volume for use in LVM volume group	pvccreate(1M)
pvddisplay - display information about physical volumes in LVM volume group	pvddisplay(1M)
pvmmove - move allocated physical extents from one LVM physical volume to other physical volumes	pvmmove(1M)
pwck - password file checker	pwck(1M)
pwconv - update secure password facility	pwconv(1M)
pwd - print current working directory	ksh(1)
pwd - print current working directory	sh-posix(1)
pwd - print working directory name	pwd(1)
pwd - working directory name	sh-bourne(1)
pwget - get password information	pwget(1)
pwgrd - password and group hashing and caching daemon	pwgrd(1M)
pwgr_stat - password and group hashing and caching statistics	pwgr_stat(1M)
qiflush() - enable/disable queue flushing	noqiflush(3X)
qsort() - quicker sort	qsort(3C)
queries and lists the status of one or more print jobs	pdq(1)
query an NIS server for information about an NIS map	yppoll(1M)
query functions for terminal insert and delay capability	has_ic(3X)
query name servers interactively	nslookup(1)
query numeric formatting conventions of current locale	localeconv(3C)
query program, Network Time Protocol	ntp(1M)
query RIP gateways	ripquery(1M)
query stream configuration	strchg(1M)
query the Name Service Switch backend libraries	nsquery(1)
query the terminfo database	tput(1)
query, set configuration and loadable flags for a module	kmsystem(1M)
query, set, or reset system parameter	kmtune(1M)
queue a signal to a process	sigqueue(2)

Description	Entry Name(Section)
queue description file for at , batch , and crontab	queuedefs(4)
queue flushing, enable/disable	noqiflush(3X)
queue, input, push a character onto	ungetch(3X)
queue, insert or remove an element	insque(3C)
queue, prints the mail queue	mailq(1)
queuedefs - queue description file for at , batch , and crontab	queuedefs(4)
quick batch mail interface	fastmail(1)
quicker sort	qsort(3C)
quit signal	glossary(9)
quot - summarize file system ownership	quot(1M)
quota consistency checker, generic file system	quotacheck(1M)
quota consistency checker, hfs file system	quotacheck_nfs(1M)
quota consistency checker, VxFS file system	quotacheck_vxfs(1M)
quota - disk quotas	quota(5)
quota - display disk usage and limits	quota(1)
quota server, remote	rquotad(1M)
quota status of specified file system, determine disk	fsclean(1M)
quotacheck - generic file system quota consistency checker	quotacheck(1M)
quotacheck_nfs - hfs file system quota consistency checker	quotacheck_nfs(1M)
quotacheck_vxfs - VxFS file system quota consistency checker	quotacheck_vxfs(1M)
quotactl(1) - manipulate disk quotas	quotactl(2)
quotaoff - turn file system quotas off	quotaon(1M)
quotaon - turn file system quotas on	quotaon(1M)
quotas, edit user	edquota(1M)
quotas, file system, turn on and off	quotaon(1M)
quotas, manipulate disk	quotactl(2)
quotas, summarize for a file system	repquota(1M)
quotient, remainder function with	remquo(3M)
quot_vxfs - summarize file system ownership	quot_vxfs(1M)
radix character	glossary(9)
radix-independent exponent	logb(3M)
radix-independent floating-point number, load exponent of	scalb(3M)
radix-independent floating-point number, load exponent of	scalbn(3M)
raise a software signal	ssignal(3C)
raise() - send signal to executing program	kill(2)
rand() - generate successive random numbers	rand(3C)
random archive member access for ELF files	elf_rand(3E)
random inode generation numbers, install	fsrand(1M)
random number generation functions	random(3M)
random() , srandom() , initstate() , setstate() - generate a pseudorandom number	random(3M)
random-number generator, simple	rand(3C)
ranlib - regenerate archive symbol table	ranlib(1)
rarpd - Reverse Address Resolution Protocol client	rarpd(1M)
rarpd - Reverse Address Resolution Protocol daemon	rarpd(1M)
raster display frame-buffer device access	framebuf(7)
raster frame-buffer display device access	framebuf(7)
rate of I/O data transfer, inform system of required minimum	io_speed_ctl(3I)
Ratfor macro processor	m4(1)
raw disk	glossary(9)
raw() - input mode control functions	cbreak(3X)
rbootd - remote (diskless) boot server	rbootd(1M)
rc - general purpose sequencer invoked upon entering new run level	rc(1M)
rcsequencer , invoked upon entering new run level	rc(1M)
rc.config - file containing system configuration information	rc.config(4)
rc.config.d - the location of files containing system configuration variable assignments	rc.config(4)
rcancel - remove requests from line printer spooling queue on remote system	rcancel(1M)
rcmd() - execute a command on a remote host	rcmd(3N)
rcp - remote file copy	rcp(1)
rcp - remote file copy	rcp(1)
rcs - change RCS file attributes	rcs(1)
RCS file attributes, change	rcs(1)

Index

All Volumes

Description	Entry Name(Section)
RCS file format	rcsfile(4)
RCS revisions, merge	rscmerge(1)
RCS: change RCS file attributes	rsc(1)
RCS: check in RCS revisions	ci(1)
RCS: check out RCS revisions	co(1)
RCS: compare RCS revisions	rscdiff(1)
RCS: description of commands	rscintro(5)
RCS: identify files in Revision Control System	ident(1)
RCS: merge RCS revisions	rscmerge(1)
RCS: print log messages and other information about RCS files	rlog(1)
rscdiff - compare RCS revisions	rscdiff(1)
rscfile - format of RCS file	rscfile(4)
rscintro - description of RCS commands	rscintro(5)
rscmerge - merge RCS revisions	rscmerge(1)
rdist - remote file distribution program	rdist(1)
rdpd - router discovery protocol daemon	rdpd(1)
rdump - incremental file system dump (for backups)	dump(1M)
re-open or open a stream file; convert file to stream	fopen(3S)
read a line from standard input	read(1)
read a POSIX semaphore	sem_getvalue(2)
read from crash dump	cr_read(3)
read from stream file or character string with formatted input conversion	scanf(3S)
read - input and parse a line	ksh(1)
read - input and parse a line	sh-posix(1)
read mail or send mail to users	mail(1)
read one line from user input	line(1)
read password from terminal while suppressing echo	getpass(3C)
read - read a line from standard input	read(1)
read - read line from standard input	sh-bourne(1)
read real-time priority	rtprio(2)
read stream up to next delimiter	bgets(3G)
read termination character on special file, set up I/O	io_eol_ctl(3I)
read text string from message file	gettext(3C)
read value of a symbolic link	readlink(2)
read() - read contiguous data from a file	read(2)
read, enable or disable block during	nodelay(3X)
read, I/O, determine how last terminated	io_get_term_reason(3I)
read, start asynchronous	aio_read(2)
read-only file system	glossary(9)
read-write lock attribute object, initialize or destroy	pthread_rwlockattr_init(3T)
read-write lock for reading, lock or attempt to lock	pthread_rwlock_rdlock(3T)
read-write lock for writing, lock or attempt to lock	pthread_rwlock_wrlock(3T)
read-write lock, initialize or destroy	pthread_rwlock_init(3T)
read-write lock, unlock	pthread_rwlock_unlock(3T)
read/write file pointer, move	lseek(2)
readable format, dump iconv translation tables to	dmpxlt(1)
readdir() - get pointer to current entry in open directory	directory(3C)
reading, open file for	open(2)
readlink() - read value of a symbolic link	readlink(2)
readmail - read mail from specified mail folder	readmail(1)
readonly - mark <i>name</i> as read-only	sh-bourne(1)
readonly - mark names as unreddefinable	ksh(1)
readonly - mark names as unreddefinable	sh-posix(1)
reads and writes: STREAMS module	tirdwr(7)
readv() - read noncontiguous data from a file	read(2)
real and effective user IDs, set	setreuid(2)
real group ID	glossary(9)
real or effective user or group ID, get	getuid(2)
real user ID	glossary(9)
real, effective, and/or saved user or group IDs, set	setresuid(2)
real-time priority, execute process with	rtprio(1)

Description	Entry Name(Section)
real-time scheduling operations	rtsched(2)
realloc() - change size of allocated memory block	malloc(3C)
realpath() - resolve pathname	realpath(3X)
realtime, POSIX extensions	aio(5)
realtime, POSIX extensions	aio_cancel(2)
realtime, POSIX extensions	aio_error(2)
realtime, POSIX extensions	aio_fsync(2)
realtime, POSIX extensions	aio_read(2)
realtime, POSIX extensions	aio_return(2)
realtime, POSIX extensions	aio_suspend(2)
realtime, POSIX extensions	aio_write(2)
realtime, POSIX extensions	lio_listio(2)
realtime, POSIX extensions	mlock(2)
realtime, POSIX extensions	mlockall(2)
realtime, POSIX extensions	munlock(2)
realtime, POSIX extensions	munlockall(2)
reblock, convert, translate and copy a (tape) file	dd(1)
reboot - reboot the system	reboot(1M)
reboot system automatically after shutting system down	shutdown(1M)
reboot system	boot(1M)
reboot() - boot the system	reboot(2)
reboots, evaluate time between	last(1)
rebuild Network Information Service database, create or	ypmake(1M)
rebuilds the database for the mail aliases file	newaliases(1M)
receipt of a signal, define what to do upon	signal(2)
receive a message from a message queue	mq_receive(2)
receive error messages from the STREAMS log driver	strerr(1M)
receive message from a socket	recv(2)
receive message from message queue	msgop(2)
recognized login shells, list of	shells(4)
reconfigure, unconfigure, configure installed software	swconfig(1M)
record displayed CRT terminal output simultaneously in a file	script(1)
record file, session, created by acctcon1 , print	acctsh(1M)
record locking and semaphores on files, provide	lockf(2)
record, audit, write for self-auditing process	audwrite(2)
records or files, move magnetic tape forward or backward by	mt(1)
recover disk space	freedisk(1M)
recover files selectively from backup media	frecover(1M)
recursively descend a directory hierarchy, executing a function	ftw(3C)
recursively expands the sendmail aliases	expand_alias(1)
recv() - receive message from a socket	recv(2)
recvfrom() - receive message from a socket	recv(2)
recvmsg() - receive message from a socket	recv(2)
red - restricted line-oriented text editor	ed(1)
redefine default login shell	chsh(1)
redefine environment for command execution	env(1)
redrawwin() - line update status functions	redrawwin(3X)
reduce LVM volume group by removing physical volumes	vgreduce(1M)
reduce multiple adjacent blank lines to single blank line	ssp(1)
reference manual, introduction	Introduction(9)
reference pages, macro package for formatting	man(5)
reformat and copy a (tape) file	dd(1)
reformat or change a text file	newform(1)
refresh control function, for window	touchwin(3X)
refresh control functions for window	is_linetouched(3X)
refresh the pad immediately after writing a character rendition	pechochar(3X)
refresh the window immediately after writing a complex character	echo_wchar(3X)
refresh() - refresh windows and lines	doupdate(3X)
refresh, determine whether a screen has been refreshed	isendwin(3X)
regcmp() - compile a regular expression	regcmp(3X)
regcomp() - regular expression matching routines	regcomp(3C)

Index All Volumes

Description

Entry Name(Section)

regenerate, maintain, and update groups of programs	make(1)
regerror() - regular expression matching routines	regcomp(3C)
regex() - execute a regular expression against a string	regcmp(3X)
regexec() - regular expression matching routines	regcomp(3C)
regfree() - regular expression matching routines	regcomp(3C)
region, initialize semaphore in mapped file or anonymous memory	msem_init(2)
region, remove semaphore in mapped file or anonymous	msem_remove(2)
region, unmap a mapped	munmap(2)
regions, first locations beyond allocated program	end(3C)
register a function to be called at program termination	atexit(2)
register fork handler	pthread_atfork(3T)
register or remove a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
register or unregister loadable kernel modules with the running kernel	kmmodreg(1M)
register/cancel a notification request with a message queue	mq_notify(2)
register_rpc() - obsolete library routines for RPC	rpc_soc(3N)
regular expression and pattern matching notation definitions	regex(5)
regular expression compile and match routines	regex(3X)
regular expression	glossary(9)
regular expression matching routines	regcomp(3C)
regular expression, compile or execute against a string	regcmp(3X)
regular expressions, compile and execute	re_comp(3X)
regular file	glossary(9)
rehash - recompute internal hash table	csh(1)
reinitialize (erase) mass storage media (use -r option)	mediainit(1)
reject - prevent LP printer queuing requests	accept(1M)
reject/select lines common to two sorted files	comm(1)
relational database, join two relations in	join(1)
relative path name	glossary(9)
relative window creation function	derwin(3X)
relatively arm a per-process timer	reltimer(3C)
release blocked signals and atomically wait for interrupt	sigpause(2)
release level of operating system, display	uname(1)
release level of operating system, get	uname(2)
release of operating system, display	uname(1)
release of operating system, get	uname(2)
reload window from a file	getwin(3X)
relocatable file, generate from object files and libraries	ld(1)
reltimer() - relatively arm a per-process timer	reltimer(3C)
remainder function	remainder(3M)
remainder function with quotient	remquo(3M)
remainder functions	fmod(3M)
remainder() - remainder function	remainder(3M)
remainder, integer division and	div(3C)
remexportent() - access exported file system information	exportent(3N)
remind you when you have to leave	leave(1)
reminder service	calendar(1)
remote (diskless) boot server	rbootd(1M)
remote backup over network	dump(1M)
remote command, return a stream to	rcmd(3N)
remote command, return stream to a	rexec(3N)
remote commands, return streams to (parallel)	prcmd(3N)
Remote Enable line on HP-IB, control the	hpib_ren_ctl(3I)
remote execution server	rexecd(1M)
remote execution server, RPC-based	rex(1M)
remote file copy	rcp(1)
remote file copy	rcp(1)
remote file distribution	rdist(1)
remote file system, perform Network File System mount to	vhe_u_mnt(1M)
remote file system, restore incrementally	rvxrestore(1M)
remote host, execute a command on a	rcmd(3N)
remote host, execute command on a	on(1)

Description	Entry Name(Section)
remote hosts and users, authorizing access on local host	hosts.equiv(4)
remote incremental file system dump (for backups)	dump(1M)
remote incremental file system restore	restore(1M)
remote kernel, get performance data from	rstat(3N)
remote login	rlogin(1)
remote login server	rlogind(1M)
remote machines, return information about users on	rnusers(3N)
remote machines, write to specified	rwall(3N)
remote magnetic tape dump and restore protocol module	rmt(1M)
remote mounts, show all	showmount(1M)
remote node, get file handle for file on	getfh(2)
remote quota server	rquotad(1M)
remote shell server	remshd(1M)
remote shell, execute from a	remsh(1)
remote system over LAN, log in on a	vt(1)
remote system, send LP request to	rlp(1M)
remote systems, cancel LP spooling requests sent to	rcancel(1M)
remote systems, daemon for LP requests from	rlpdaemon(1M)
remote terminal, spawn getty to (call terminal)	ct(1)
remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
remote user communication server	talkd(1M)
remote user information server	fingerd(1M)
remote user, verify as a local user	rcmd(3N)
remote uucp or uux command requests, execute on local system	uuxqt(1M)
remotely mounted file systems, keep track of	mount(3N)
remove a delta from an SCCS file	rmDEL(1)
remove a directory file	rmdir(2)
remove a file or directory	rm(1)
remove a LIF file	lifrm(1)
remove a message queue, semaphore set, or shared memory identifier	ipcrm(1)
remove a printer for use with tsm , add	tsm.lpadmin(1M)
remove a special (device) file	rmsf(1M)
remove all blank lines from file	rmnl(1)
remove and unconfigure software products	swremove(1M)
remove boot programs from a disk device, install, update, or	mkboot(1M)
remove directory entry	unlink(2)
remove directory	rmdir(1)
remove DOS files or directories	dosrm(1)
remove extra new-line characters from file	rmnl(1)
remove file that is not listed in any directory	clri(1M)
remove logical volumes from LVM volume group	lvremove(1M)
remove LVM logical volume link to root, primary swap, or dump volume	lvrmboot(1M)
remove LVM volume group definition from the system	vgremove(1M)
remove multiple line-feeds from output	ssp(1)
remove NIS+ directories	nisrmdir(1)
remove NIS+ objects from the namespace	nisrm(1)
remove nroff / troff , tbl , and neqn constructs	deroff(1)
remove or insert an element in a queue	insque(3C)
remove or register a thread cancellation cleanup handler	pthread_cleanup_pop(3T)
remove outdated STREAMS error log files	strclean(1M)
remove physical volumes from an LVM volume group	vgreduce(1M)
remove preprocessor lines	unifdef(1)
remove reverse line-feeds and backspaces from text	col(1)
remove semaphore in mapped file or anonymous region	msem_remove(2)
remove symbol and line number information from an object file	strip(1)
remove user crontab file	crontab(1)
remove() - remove a file	remove(3C)
remove, create directories in a path	mkdirp(3G)
removes all jobs from the specified object	pdclean(1)
removes print jobs	pdrn(1)
remque() - remove an element in a queue	insque(3C)

Index All Volumes

Description	Entry Name(Section)
remquo() - remainder function with quotient	remquo(3M)
remsh - execute from a remote shell	remsh(1)
remshd - remote shell server	remshd(1M)
rename directory	mv(1)
rename file	mv(1)
rename LIF files	lifrename(1)
rename() - change the name of a file	rename(2)
renditions and characters, draw lines from single-byte	hline(3X)
renditions of characters in a window, change	chgat(3X)
renice - alter priority of running processes	renice(1)
reorganize a VxFS file system	fsadm_vxfs(1M)
repair damaged file system (generic)	fsdb(1M)
repair damaged HFS file system	fsdb_hfs(1M)
repair file system interactively and check consistency	fsck(1M)
repair file system interactively and check consistency (generic)	fsck(1M)
repair or check a physical volume in LVM volume group	pvck(1M)
repair VxFS file system	fsck_vxfs(1M)
repeat - execute command more than once	csh(1)
repeated (adjacent) lines in a file, count, extract, or eliminate	uniq(1)
repetitively affirmative responses	yes(1)
replace selected characters	tr(1)
report adjacent repeated lines in a file	uniq(1)
report CPU time used	clock(3C)
report daily system activity	sa1(1M)
report disk usage	du(1)
report I/O statistics	iostat(1)
report NIS+ server statistics	nisstat(1M)
report number of free CDFS, HFS, or NFS file system disk blocks	df_hfs(1M)
report number of free disk blocks on a VxFS file system	df_vxfs(1M)
report number of free DOS disk clusters	dosdf(1)
report number of free file system disk blocks (generic)	df(1M)
report process status	ps(1)
report RPC information	rpcinfo(1M)
report status of interprocess communication facilities	ipcs(1)
report virtual memory statistics	vmstat(1)
reporter, system activity	sar(1M)
reposition or get pointer for I/O operations on a stream file	fseek(3S)
repquota - summarize file system quotas	repquota(1M)
request server, PFS mount	pfs_mountd(1M)
request, receive confirmation (X/OPEN TLI-XTI)	t_rcvconnect(3)
requested status condition becomes true, wait until the	hpib_status_wait(3I)
requests on an LP printer	lp(1)
requests, daemon that responds to SNMP	snmpd(1M)
requests, LP	see LP requests
requests, LP spooling, move to a specified destination	lpsched(1M)
required minimum I/O data transfer rate, inform system of	io_speed_ctl(3I)
reserve a line for a dedicated purpose	riponline(3X)
reserve disk space	prealloc(1)
reserved port socket, return a	rcmd(3N)
reset an I/O interface	io_reset(3I)
reset FDDI interface, stop and	fddistop(1M)
reset, set, or query system parameter	kmtune(1M)
resetty() - save/restore terminal mode	resetty(3X)
reset_prog_mode() - restore shell terminal modes to "program" state	def_prog_mode(3X)
reset_shell_mode() - restore terminal modes to "shell" state	def_prog_mode(3X)
resize or reorganize a VxFS file system	fsadm_vxfs(1M)
resolution protocol, address	arp(7P)
resolve pathname	realpath(3X)
resolver configuration file	resolver(4)
resolver routines	resolver(3N)
resource consumption limit, get or set system	getrlimit(2)

Description	Entry Name(Section)
resource utilization, get information	getrusage(2)
respond to vt requests from other systems	vtdaemon(1M)
response to parallel poll on HP-IB, control	hpib_card_ppoll_resp(3I)
response, ask for user response for SD-UX	swask(1M)
responses, repetitively affirmative	yes(1)
restartterm() - interface to terminfo database	del_curterm(3X)
restore file system incrementally	restore(1M)
restore file system incrementally, local or across network	vxrestore(1M)
restore - incrementally restore file system	restore(1M)
restore or save file position indicator for a stream	fgetpos(3S)
restore shell terminal modes to "program" state	def_prog_mode(3X)
restore signal action	sighold(2V)
restore signal action	sigset(3C)
restore terminal mode	resetty(3X)
restore terminal modes to "shell" state	def_prog_mode(3X)
restore volume group configuration	vgcfgrestore(1M)
restore/save stack environment for non-local goto	setjmp(3C)
restricted mailer (send only)	mail(1)
restricted shell for sendmail	smrsh(1M)
restricted window attribute control functions	attroff(3X)
resubmits previously submitted print jobs	pdresubmit(1)
resume execution of a thread	pthread_resume_np(3T)
resume or suspend auditing on current process	audswitch(2)
resvport() - return a reserved port socket	rcmd(3N)
res_init() - resolver routines	resolver(3N)
res_mkquery() - resolver routines	resolver(3N)
res_query() - resolver routines	resolver(3N)
res_search() - resolver routines	resolver(3N)
res_send() - resolver routines	resolver(3N)
retrieve and set the current thread's cancelability state and type	pthread_setcancelstate(3T)
retrieve archive member header for ELF files	elf_getarhdr(3E)
retrieve archive symbol table for ELF files	elf_getarsym(3E)
retrieve capabilities from the terminfo database	tigetflag(3X)
retrieve class-dependent object file header for ELF files	elf_getehdr(3E)
retrieve class-dependent program header table for ELF files	elf_getphdr(3E)
retrieve class-dependent section header for ELF files	elf_getshdr(3E)
retrieve crash dump information	cr_info(3)
retrieve file identification data for ELF files	elf_getident(3E)
retrieve information on loaded module (program or shared library)	dlget(3C)
retrieve information on loaded module (program or shared library)	dlmodinfo(3C)
retrieve name of load module	dlgetname(3C)
retrieve SD product from new SD media	swgettools(1M)
retrieve uninterpreted file contents for ELF files	elf_rawfile(3E)
return a reserved port socket	rcmd(3N)
return a stream to a remote command	rcmd(3N)
return asynchronous I/O status	aio_return(2)
return character back into input stream	ungetc(3S)
return - exit function with return value	sh-bourne(1)
return how many processors are installed in the system	pthread_processor_bind_np(3T)
return HP-UX process priority	rtsched(2)
return information about users on remote machines	rnusers(3N)
return integer absolute value	abs(3C)
return maximum for scheduling policy	rtsched(2)
return minimum for scheduling policy	rtsched(2)
return network status	netstat(1)
return POSIX process priority	rtsched(2)
return scheduling parameters	rtsched(2)
return scheduling policy	rtsched(2)
return - shell function return to invoking script	ksh(1)
return - shell function return to invoking script	sh-posix(1)
return status of HP-IB interface	hpib_bus_status(3I)

Index All Volumes

Description

Entry Name(Section)

return stream to a remote command	rexec(3N)
return the size of an object file type for elf32 or elf64 files	elf_fsize(3E)
return the state of the NIS+ namespace using a conditional expression	nistest(1)
return wide character back into input stream	ungetwc(3C)
rev - reverse the text character sequence in each line of a file	rev(1)
revck - check internal revision numbers of HP-UX files	revck(1M)
Reverse Address Resolution Protocol client	rarpd(1M)
Reverse Address Resolution Protocol daemon	rarpd(1M)
reverse line-feeds and backspaces, remove from text	col(1)
reverse order, show last commands executed in	lastcomm(1)
reverse - reverse printer pages for collating	lpfilter(1)
reverse the left-to-right text character sequence in each line of a file	rev(1)
Revision Control System	see RCS
revision numbers of HP-UX files, verify internal	revck(1M)
revisions, compare RCS	rcsdiff(1)
revisions, RCS, check in	ci(1)
revisions, RCS, check out	co(1)
rewind legal user shells file	getusershell(3C)
rewind magnetic tape	mt(1)
rewind() - set position of next I/O operation on stream file	fseek(3S)
rewinddir() - reset position of named directory stream to beginning of directory	directory(3C)
rewind_unlocked() - set position of next I/O operation on stream file, no locking of stream for multi-thread applications	fseek(3S)
rewrite an existing file	creat(2)
rexcd - RPC-based remote execution server	rexcd(1M)
rexec() , login information for	netrc(4)
rexec() - return stream to a remote command	rexec(3N)
rexecd - remote execution server	rexecd(1M)
re_comp() - compile and execute regular expressions	re_comp(3X)
re_exec() - compile and execute regular expressions	re_comp(3X)
right or left justify lines for NLS printing	nljust(1)
right triangle, hypotenuse of a	hypot(3M)
rights to a file, get a user's effective access	getaccess(2)
rights, access, to file(s), list	getaccess(1)
rindex() - BSD portability string routine	string(3C)
ring, list characteristics of nodes on FDDI	fddinet(1M)
rint() - round to nearest int function	rint(3M)
RIP gateways, query	ripquery(1M)
ripoffline() - reserve a line for a dedicated purpose	ripoffline(3X)
ripquery - query RIP gateways	ripquery(1M)
rksh - restricted Korn shell command programming language	ksh(1)
rlog - print log messages and other information about RCS files	rlog(1)
rlogin - remote login	rlogin(1)
rlogind - remote login server	rlogind(1M)
rlp - send LP line printer request to a remote system	rlp(1M)
rlpdaemon - line printer daemon for LP requests from remote systems	rlpdaemon(1M)
rlpstat - print status of LP requests sent to remote system	rlpstat(1M)
rm - remove files or directories	rm(1)
rmail - restricted mailer (send only)	mail(1)
rmboot - remove boot programs from a disk device	mkboot(1M)
rmDEL - remove a delta from an SCCS file	rmDEL(1)
rmdir - remove directories	rmdir(1)
rmdir() - remove a directory file	rmdir(2)
rmdirp() - remove directories in a path	mkdirp(3G)
rmnl - remove extra new-line characters from file	rmnl(1)
rmsf - remove a special (device) file	rmsf(1M)
rmt - remote magnetic tape protocol module	rmt(1M)
rmtab - local file system mount statistics	rmtab(4)
rmtimer() - free a per-process timer	rmtimer(3C)
rnusers() - return information about users on remote machines	rnusers(3N)
root directory	glossary(9)

Description	Entry Name(Section)
root directory, change	chroot(2)
root directory, change for a command	chroot(1M)
root file system, prepare for migration from partitions to LVM logical volumes	lvmmigrate(1M)
root volume	glossary(9)
root volume, prepare LVM logical volume to be	lvlnboot(1M)
root volume, remove LVM logical volume link	lvrmboot(1M)
root, target, modify software products in depot or	swmodify(1M)
roots and depots, register or unregister	swreg(1M)
round function	round(3M)
round to long function	lround(3M)
round to long long function	llround(3M)
round to nearest int functions	rint(3M)
round to nearest long function	lrint(3M)
round to nearest long long function	llrint(3M)
round() - round function	round(3M)
rounding mode: getting floating-point	fegetround(3M)
rounding mode: setting floating-point	fesetround(3M)
route and path between hosts, compute shortest	pathalias(1)
route - manipulate routing tables manually	route(1M)
router connection mapper, multicast	map-mbone(1M)
router, electronic address	pathalias(1)
routine for manipulating global RPC attribute for client and server applications	rpc_control(3M)
routine for sorted tables, binary search	bsearch(3C)
routine to retrieve user name, PAM	pam_get_user(3)
routines for client side calls	rpc_clnt_calls(3N)
routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
routines for dealing with creation and manipulation of CLIENT handles	rpc_clnt_create(3N)
routines for external data representation	xdr(3N)
routines for external data representation	xdr_admin(3N)
routines for external data representation	xdr_complex(3N)
routines for external data representation	xdr_simple(3N)
routines for external data representation stream creation	xdr_create(3N)
routines for PAM, authentication information	pam_set_item(3)
routines for registering servers, rpc	rpc_svc_reg(3N)
routines for remote procedure calls, rpc	rpc(3N)
routines for remote procedure calls, XDR	rpc_xdr(3N)
routines for RPC servers	rpc_svc_calls(3N)
routines for secure remote procedure calls	secure_rpc(3N)
routines for server side remote procedure call errors	rpc_svc_err(3N)
routines for the creation of server handles, rpc	rpc_svc_create(3N)
routines to maintain module specific state, PAM	pam_set_data(3)
routines, authentication transaction routines for PAM	pam_start(3)
routines, define label for formatting	setlabel(3)
routines, emulate /etc/termcap access	termcap(3X)
routines, Internet address manipulation	inet(3N)
routines, library routines for RPC bind service	rpcbind(3N)
routines, network station address string conversion	net_aton(3C)
routines, obsolete library routines for RPC	rpc_soc(3N)
routines, resolver	resolver(3N)
routing daemon, gateway	gated(1M)
routing daemon, IP multicast	mrouted(1M)
routing - system support for local network packet routing	routing(7)
routing tables manually, manipulate	route(1M)
routing, multicast, configuration information tool	mrinfo(1M)
RPC entry, get	getrpcent(3C)
RPC information, report	rpcinfo(1M)
rpc - library routines for remote procedure calls	rpc(3N)
RPC port number, get	getrpcport(3N)
RPC program number mapper, universal addresses to	rpcbind(1M)
RPC protocols, generate C header files	rpcgen(1)
rpc RPC program number database	rpc(4)

Description

Entry Name(Section)

rpc, CLIENT handles, library routines for dealing with creation and manipulation of	rpc_clnt_create(3N)
rpc, library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
rpc, library routines for client side calls	rpc_clnt_calls(3N)
rpc, library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
rpc, library routines for registering servers	rpc_svc_reg(3N)
rpc, library routines for remote procedure calls	rpc(3N)
rpc, library routines for RPC bind service	rpcbind(3N)
rpc, library routines for RPC servers	rpc_svc_calls(3N)
rpc, library routines for secure remote procedure calls	secure_rpc(3N)
rpc, library routines for the creation of server handles	rpc_svc_create(3N)
rpc, obsolete library routines for RPC	rpc_soc(3N)
RPC, secure, change user's key	chkey(1)
RPC-based remote execution server	rexnd(1M)
rpc.nisd - NIS+ service daemon	rpc.nisd(1M)
rpc.nisd_resolv - NIS+ service daemon	rpc.nisd(1M)
rpc.nispasswd() - NIS+ password update daemon	rpc.nispasswd(1M)
rpc.pcnfsd - PC-NFS authentication and print request server	pcnfsd(1M)
rpc.yupdated, yupdated, - server for changing NIS information	yupdated(1M)
rpcbind - universal addresses to RPC program number mapper	rpcbind(1M)
rpcbind() - library routines for RPC bind service	rpcbind(3N)
rpcb_getaddr() - library routines for RPC bind service	rpcbind(3N)
rpcb_getmaps() - library routines for RPC bind service	rpcbind(3N)
rpcb_gettime() - library routines for RPC bind service	rpcbind(3N)
rpcb_rmtcall() - library routines for RPC bind service	rpcbind(3N)
rpcb_set() - library routines for RPC bind service	rpcbind(3N)
rpcb_unset() - library routines for RPC bind service	rpcbind(3N)
rpcgen - generate RPC protocols, C header files	rpcgen(1)
rpcinfo - report RPC information	rpcinfo(1M)
rpc_broadcast() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_broadcast_exp() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_call() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_clnt_auth() - library routines for client side remote procedure call authentication	rpc_clnt_auth(3N)
rpc_clnt_calls() - library routines for client side calls	rpc_clnt_calls(3N)
rpc_clnt_create() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
rpc_control() - library routine for manipulating global RPC attribute for client and server applications	rpc_control(3N)
rpc_createerr() - library routines for dealing with CLIENT handles	rpc_clnt_create(3N)
rpc_reg() - library routines for registering servers	rpc_svc_reg(3N)
rpc_soc() - obsolete library routines for RPC	rpc_soc(3N)
rpc_svc_calls() - library routines for RPC servers	rpc_svc_calls(3N)
rpc_svc_create() - library routines for the creation of server handles	rpc_svc_create(3N)
rpc_svc_err() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
rpc_svc_reg() - library routines for registering servers	rpc_svc_reg(3N)
rpc_xdr() - XDR library routines for remote procedure calls	rpc_xdr(3N)
rpr - repair parity information on an HP SCSI disk array LUN	rpr(1M)
rquotad - remote quota server	rquotad(1M)
RR scheduling policy	rtsched(2)
RR2 scheduling policy	rtsched(2)
rrestore - incrementally restore file system across network	restore(1M)
rsh - restricted shell command programming language	sh-bourne(1)
rsh - rshell, the restricted command programming language	sh-bourne(1)
rstat() - get performance data from remote kernel	rstat(3N)
rstatd - kernel statistics server	rstatd(1M)
rtprio - execute process with real-time priority	rtprio(1)
RTPRIO scheduling policy	rtsched(2)
rtprio() - change or read real-time priority	rtprio(2)
rtsched - execute process with POSIX real-time priority	rtsched(1)
rtsched - real time scheduling operations	rtsched(2)
run a command at nondefault priority	nice(1)
run a command immune to hangups	nohup(1)
run daily accounting	runacct(1M)

Description	Entry Name(Section)
run level	init(1M)
run-level s, place system in	shutdown(1M)
runacct - accumulate accounting data and command usage summary	acctsh(1M)
runacct - run daily accounting	runacct(1M)
running processes, alter priority of	renice(1)
rup: show host status of local machines (RPC version)	rup(1)
ruptime - show status of local machines	ruptime(1)
ruserok() - verify a remote user as a local user	rcmd(3N)
rusers: determine who is logged in on local network machines	rusers(1)
rusers(): return information about users on remote machines	rnusers(3N)
rusersd - network username server	rusersd(1M)
rvxdump - incremental file system dump across network	vxdump(1M)
rvxrestore - restore file system incrementally across network	vxrestore(1M)
rwall server, network	rwall(1M)
rwall - write to all users over a network	rwall(1M)
rwall(): write to specified remote machines	rwall(3N)
rwalld - network rwall server	rwalld(1M)
rwho - show who is logged in on local machines	rwho(1)
rwhod - system status server	rwhod(1M)
sa1, sa2, sadc - system activity report package	sa1(1M)
sa1 - collect and output or store system activity data in binary file	sa1(1M)
sa2 - write daily system activity report in binary file	sa1(1M)
sact - print current SCCS file editing activity	sact(1)
sad - STREAMS Administrative Driver	sad(7)
sadc - collect and output or store system activity data	sa1(1M)
SAM logfile, tool for viewing and saving	samlog_viewer
sam - system administration manager	sam(1M)
samlog_viewer - tool for viewing and saving the SAM logfile	samlog_viewer(1)
sar - system activity reporter	sar(1M)
save a crash dump of the operating system	savecrash(1M)
save or restore file position indicator for a stream	fgetpos(3S)
save or restore program or shell terminal modes	def_prog_mode(3X)
save terminal modes as the "shell" state	def_prog_mode(3X)
save/restore stack environment for non-local goto	setjmp(3C)
save/restore terminal mode	resetty(3X)
savecrash - save a crash dump of the operating system	savecrash(1M)
saved group ID	glossary(9)
saved process group ID	glossary(9)
saved set-group-ID	glossary(9)
saved set-user-ID	glossary(9)
saved user ID	glossary(9)
saved, real, and/or effective user or group IDs, set	setresuid(2)
saveetty() - save/restore terminal mode	resetty(3X)
saving, viewing SAM logfile tool	samlog_viewer
sbrk() - increase data segment space allocation	brk(2)
scalb() - load exponent of a radix-independent floating-point number	scalb(3M)
scalbn() - load exponent of a radix-independent floating-point number	scalbn(3M)
scan a directory	scandir(3C)
scan physical volumes for LVM volume groups	vgscan(1M)
scan the I/O system for disk arrays	arrayscan(1M)
scan the I/O system	ioscan(1M)
scandir() - scan a directory	scandir(3C)
scanf() - formatted read from standard input stream file	scanf(3S)
scanner, big file	bfs(1)
scanning and processing language, text pattern	awk(1)
scanw() - convert formatted input from a window	mvscanw(3X)
scatter data to check the network	spray(3N)
SCCS (Source Code Control System)	glossary(9)
SCCS command help	scshelp(1)
SCCS commands, utility program for	scs(1)
SCCS delta, change delta commentary of	cdc(1)

Index

All Volumes

Description	Entry Name(Section)
SCCS deltas, combine	comb(1)
SCCS file format	sccsfile(4)
SCCS file	glossary(9)
sccs - utility program for SCCS commands	sccs(1)
SCCS, create and administer SCCS files	admin(1)
SCCS: combine SCCS deltas	comb(1)
SCCS: compare two versions of an SCCS file	sccsdiff(1)
SCCS: get a version of an SCCS file	get(1)
SCCS: get SCCS identification information from files	what(1)
SCCS: make a delta (change) to an SCCS file	delta(1)
SCCS: print and summarize an SCCS file	prs(1)
SCCS: print current SCCS file editing activity	sact(1)
SCCS: remove a delta from an SCCS file	rmddl(1)
SCCS: undo a previous get of an SCCS file	unget(1)
SCCS: validate an SCCS file	val(1)
sccsdiff - compare two versions of an SCCS file	sccsdiff(1)
sccsfile - format of SCCS file	sccsfile(4)
scscshelp - help for SCCS commands	scscshelp(1)
schedule uucp transport files	uusched(1M)
scheduler, LP, start or stop	lpsched(1M)
scheduling operations, real-time	rtsched(2)
scheduling policy	rtsched(2)
scheduling priority	renice(1)
SCHED_FIFO scheduling policy	rtsched(2)
sched_getparam() - return scheduling parameters	rtsched(2)
sched_getscheduler() - return scheduling policy	rtsched(2)
sched_get_priority_max() - return maximum for scheduling policy	rtsched(2)
sched_get_priority_min() - return minimum for scheduling policy	rtsched(2)
SCHED_HPUX scheduling policy	rtsched(2)
SCHED_OTHER scheduling policy	rtsched(2)
SCHED_RR scheduling policy	rtsched(2)
SCHED_RR2 scheduling policy	rtsched(2)
sched_rr_get_interval() - update execution time limit	rtsched(2)
SCHED_RTPRIO scheduling policy	rtsched(2)
sched_setparam() - set scheduling parameters	rtsched(2)
sched_setscheduler() - set scheduling policy	rtsched(2)
SCHED_TIMESHARE scheduling policy	rtsched(2)
sched_yield() - force process to relinquish processor	rtsched(2)
schgr - SCSI media changer device drivers	autochanger(7)
scn - scan HP SCSI disk array LUNs for parity consistency	scn(1M)
screen file input/output functions	scr_dump(3X)
screen initialisation functions	initscr(3X)
screen size information, specify source	use_env(3X)
screen, clear terminal	clear(1)
screen, determine if it has been refreshed	isendwin(3X)
screen, free storage associated with a screen	delscreen(3X)
screen, number of columns	COLS(3X)
screen, number of lines on	LINES(3X)
screen-oriented mail interface	elm(1)
screen-oriented text editor	vi(1)
screen: flash the screen	flash(3X)
screens, switch between	set_term(3X)
script for the init process	inittab(4)
script format and semantics, localedef -command input	localedef(4)
script - make typescript of terminal session	script(1)
script to set up user's environment at login, shell	profile(4)
scripts, symbolic translation file for localedef	charmap(4)
scr1() - scroll the window, enhanced curses	scr1(3X)
scroll a curses window	scroll(3X)
scroll the window, enhanced curses	scr1(3X)
scroll() - scroll a curses window	scroll(3X)

Description	Entry Name(Section)
scrollok() - terminal output control functions	clearok(3X)
scr_dump() - screen file input/output functions	scr_dump(3X)
scr_init() - screen file input/output functions	scr_dump(3X)
scr_restore() - screen file input/output functions	scr_dump(3X)
scr_set() - screen file input/output functions	scr_dump(3X)
SCSI device drivers	scsi(7)
SCSI device, control a	scsictl(1M)
SCSI direct access device driver	scsi_disk(7)
SCSI media changer device drivers	autochanger(7)
SCSI pass-through device driver	scsi_ctl(7)
SCSI pass-through device driver	scsi_pt(7)
SCSI sequential access device driver	scsi_tape(7)
scsictl - control a SCSI device	scsictl(1M)
scsi_ctl - SCSI pass-through device driver	scsi_ctl(7)
scsi_disk - SCSI direct access device driver	scsi_disk(7)
scsi_pt - SCSI pass-through device driver	scsi_pt(7)
scsi_tape - SCSI sequential access device driver	scsi_tape(7)
sd - create and monitor jobs	swjob(1M)
sd - create, distribute, install, monitor, and manage software	sd(5)
SD product from new SD media	swgettools(1M)
sd - SD objects, attributes, and storage formats	sd(4)
sdiff - compare two files and show differences side-by-side	sdiff(1)
search a file for a string or expression	grep(1)
search and print process accounting files	acctcom(1M)
search directory tree for files	find(1)
search environment list for value of specified variable name	getenv(3C)
search for files	find(1)
search for named file in named directories	pathfind(3G)
search path for dynamically loadable kernel modules, change	modpath(2)
search physical volumes for LVM volume groups	vgscan(1M)
search routine, binary, for sorted tables	bsearch(3C)
search table for entry; optional update if missing	lsearch(3C)
search tables, hash, manage	hsearch(3C)
search tree, manage a binary	tsearch(3C)
searching NIS+ tables	nismatch(1)
secOf2() , SECOF2() - test for valid second byte in 16-bit character	nl_tools_16(3X)
secondary prompt	glossary(9)
secret key, decrypt and store	keylogin(1)
secret key, delete key stored with keyserv	keylogout(1)
secret key, retrieve public or	getpublickey(3M)
section data for ELF files, manipulate	elf_getdata(3E)
section information for ELF files, get	elf_getscn(3E)
section sizes and allocation space of object files, print	size(1)
section sizes, disk, calculate default	disksecn(1M)
secure internet services description	sis(5)
secure internet services, configuration file	inetsvcs.conf(4)
secure internet services, enable or disable	inetsvcs_sec(1M)
secure password file, get entry from	getspwnt(3X)
secure remote procedure calls, library routines	secure_rpc(3N)
secure RPC key, change user's	chkey(1)
secured password facility, update	pwconv(1M)
securenets(4) - NIS map security file	securenets(4)
securetty file	login(1)
secure_rpc() - library routines for secure remote procedure calls	secure_rpc(3N)
security databases	authcap(4)
security file for ftpd(1M)	ftpusers(4)
security file, inetd optional	inetd.sec(4)
security file, map NIS	securenets(4)
security files authorizing access by remote hosts and users on local host	hosts.equiv(4)
security purposes, destroy mass storage data for (use -r option)	mediainit(1)
security, audio	see audio(5)

Index All Volumes

Description	Entry Name(Section)
sed - streaming text editor	sed(1)
see - access EEPROM bytes in an HP SCSI disk array controller	see(1M)
seek ; move read/write file pointer	lseek(2)
seekdir() - set position of next readdir() operation on named directory stream	directory(3C)
setreuid() - set real and effective user IDs	setreuid(2)
segment, get shared memory	shmget(2)
select code	glossary(9)
select users to audit	audusr(1M)
select() - synchronous I/O multiplexing	select(2)
select/reject lines common to two sorted files	comm(1)
selected characters, alter, delete, modify, substitute, translate	tr(1)
selected fields of each line in a file, cut out (extract)	cut(1)
selected keys in Network Information Service map, print the values of	ypmatch(1)
selectively recover files from backup media	frecover(1M)
self-auditing process, write audit record for	auditwrite(2)
semantics, localedef -command input script format and	localedef(4)
semaphore control operations	semctl(2)
semaphore identifier (semid)	glossary(9)
semaphore in mapped file or anonymous region, remove	msem_remove(2)
semaphore in mapped file or anonymous memory region, initialize	msem_init(2)
semaphore operation permissions	glossary(9)
semaphore operations	semop(2)
semaphore set identifier, remove	ipcrm(1)
semaphore, lock a	msem_lock(2)
semaphore, unlock a	msem_unlock(2)
semaphores and record locking on files, provide	lockf(2)
semaphores, get set of	semget(2)
semaphores, report status	ipcs(1)
semctl() - semaphore control operations	semctl(2)
semget() - get set of semaphores	semget(2)
semid (semaphore identifier)	glossary(9)
semop() - semaphore operations	semop(2)
sem_close() - close a named semaphore	sem_close(2)
sem_destroy() - destroy an unnamed semaphore	sem_destroy(2)
sem_getvalue() - read a POSIX semaphore	sem_getvalue(2)
sem_init() - initialize an unnamed semaphore	sem_init(2)
sem_open() - open/create a named semaphore	sem_open(2)
sem_post() - unlock a POSIX semaphore	sem_post(2)
sem_trywait() - lock a POSIX semaphore without blocking	sem_wait(2)
sem_unlink() - unlink a named semaphore	sem_unlink(2)
sem_wait() - lock a POSIX semaphore	sem_wait(2)
send a message on a stream	putmsg(2)
send a message simultaneously to all users	wall(1M)
send a message to a message queue	mq_send(2)
send a signal to a process or a group of processes	sigsend(2)
send a signal to a thread	pthread_kill(3T)
send BOOTREQUEST to BOOTP server	bootpquery(1M)
send command bytes over HP-IB	hpib_send_cmnd(3I)
send commands to Terminal Session Manager	tsm.command(1)
send copy of standard output to specified file	tee(1)
send Fibre Channel Light Weight Protocol Echo Request packet	pong(1M)
send LP line printer request to a remote system	rlp(1M)
send mail over the Internet	sendmail(1M)
send mail to users or read mail	mail(1)
send message to a socket	send(2)
send message to message queue	msgop(2)
send ping to NIS+ servers	nisping(1M)
send signal to executing program	kill(2)
send signal to process	kill(1)
send signals to domain name server	sig_named(1M)
send test packets	ping(1M)

Description	Entry Name(Section)
send the contents of a file through a socket	sendfile(2)
send() - send message to a socket	send(2)
sendfile() - send the contents of a file through a socket	sendfile(2)
sendmail aliases file	aliases(5)
sendmail aliases, print system-wide	praliases(1)
sendmail aliases, recursively expands	expand_alias(1)
sendmail daemon, killing it	killsm(1M)
sendmail database maps, creating	makemap(1M)
sendmail - send mail over the Internet	sendmail(1M)
sendmail, restricted shell	smrsh(1M)
sendmail.cf files, convert to new format	convert_awk(1M)
sendmsg() - send message to a socket	send(2)
sendto() - send message to a socket	send(2)
send_sound - play audio file	send_sound(1)
separate a file into multiple <i>n</i> -line pieces	split(1)
separate double-precision number into mantissa and exponent	frexp(3M)
separate mirrored LVM logical volume into two logical volumes	lvsplit(1M)
sequence for object code files in a library, find optimum	lorder(1)
sequential access device driver, SCSI	scsi_tape(7)
sequential archive member access for ELF files, provide	elf_next(3E)
Serial and HP AdvanceLink server, Basic	pcserver(1M)
serial modem line control, asynchronous	modem(7)
serial poll on HP-IB bus, conduct a	hpiib_spoll(3I)
serialize - force target process to run serially with other processes	serialize(1)
serialize() -force target process to run serially with other processes	serialize(2)
server and client, NIS+, initialization utility	nisinit(1M)
server for changing NIS information	yppupdated(1M)
server for information about an NIS map, query an NIS	yppoll(1M)
server for storing private encryption keys	keyserv(1)
server or map master, list which host is Network Information System	ypwhich(1)
server to local node, transfer NIS database from NIS	ypxfr(1M)
server, Basic Serial and HP AdvanceLink	pcserver(1M)
server, domain name, send signals to	sig_named(1M)
server, file transfer protocol	ftpd(1M)
server, Internet Boot Protocol	bootpd(1M)
server, Internet domain name	named(1M)
server, kernel statistics	rstatd(1M)
server, network rwall	rwalld(1M)
server, network username	rusersd(1M)
server, NFS mount request	mountd(1M)
server, NIS+, statistics, report	nisstat(1M)
server, PFS mount request	pfs_mountd(1M)
server, remote execution	rexecd(1M)
server, remote login	rlogind(1M)
server, remote quota	rquotad(1M)
server, remote shell	remshd(1M)
server, remote user information	fingerd(1M)
server, RPC-based remote execution	rex(1M)
server, send BOOTREQUEST to BOOTP	bootpquery(1M)
server, spray	sprayd(1M)
server, system status	rwhod(1M)
server, TELNET protocol	telnetd(1M)
server, trivial file transfer protocol	tftpd(1M)
servers, library routines for registering servers, rpc	rpc_svc_reg(3N)
servers, library routines for RPC servers	rpc_svc_calls(3N)
servers, library routines for server side remote procedure call errors	rpc_svc_err(3N)
servers, library routines for the creation of server handles, rpc	rpc_svc_create(3N)
servers, name, query interactively	nslookup(1)
servers, set up NIS+	nisserver(1M)
service daemon, NIS+	rpc.nisd(1M)
service entry, get, set, or end	getservent(3N)

Index

All Volumes

Description

Entry Name(Section)

service module APIs, PAM	pam_sm(3)
service module, PAM user policy definition	pam_updbe(5)
service name database	services(4)
service provider implementation for pam_acct_mgmt	pam_sm_acct_mgmt(3)
service provider implementation for pam_authenticate()	pam_sm_authenticate(3)
service provider implementation for pam_chauthtok()	pam_sm_chauthtok(3)
service provider implementation for pam_open_session() and pam_close_session()	pam_sm_open_session(3)
service provider implementation for pam_setcred()	pam_sm_setcred(3)
service switch	service.switch(1M)
service vt requests from other systems	vtdaemon(1M)
service, reminder	calendar(1)
service.switch - indicate lookup sources and fallback mechanism	service.switch(1M)
services daemon, Internet	inetd(1M)
services - service name database	services(4)
session creation and termination operations, PAM	pam_open_session(3)
session	glossary(9)
session ID, get	getsid(2)
session ID, get terminal	tcgetsid(3C)
session leader	glossary(9)
session lifetime	glossary(9)
Session Manager state information, get Terminal	tsm.info(1)
Session Manager, Terminal	tsm(1)
session record file created by acctcon1 , print	acctsh(1M)
session, authentication, account, and password management PAM modules for UNIX	pam_unix(5)
session, create and set process group ID	setsid(2)
session, record typescript of terminal output during	script(1)
session, start terminal	login(1)
set a process's alarm clock	alarm(2)
set access control list (ACL) information	setacl(2)
set access permissions mode mask for file-creation	umask(1)
set and clear window attributes	standend(3X)
set and get concurrency level of unbound threads	pthread_getconcurrency(3T)
set and get current user context	getcontext(2)
set and get the scheduling policy and associated parameters	pthread_getschedparam(3T)
set and get the thread-specific data associated with a key	pthread_getspecific(3T)
set and retrieve the current thread's cancelability state and type	pthread_setcancelstate(3T)
set and/or get signal alternate stack context	sigaltstack(2)
set and/or get signal stack context	sigstack(2)
set attributes for pthread	pthread_attr_getdetachstate(3T)
set audit ID (aid()) for current process	setaudit(2)
set cchar_t from a wide character string and rendition	setcchar(3X)
set contents of memory area to specified byte	memory(3C)
set current ignorable signals mask	sigsetmask(2)
set current signal mask	sigsetmask(2)
set current system-clock date and time to new value	date(1)
set environment for command execution	env(1)
set extent attributes (VxFS)	setext(1M)
set file access and modification times	utime(2)
set file access and modification times	utimes(2)
set fill byte for ELF files	elf_fill(3E)
set foreground process group ID	tcsetpgrp(3C)
set group access list	setgroups(2)
set LVM volume group availability	vgchange(1M)
set name of current host system	hostname(1)
set name of current NIS domain	getdomainname(2)
set name of host cpu	sethostname(2)
set network entry	getnetent(3N)
set network group entry	getnetgrent(3C)
set network host entry	gethostent(3N)
set Network Information Service domain name	domainname(1)

Description	Entry Name(Section)
set node name	uname(1)
set node name	uname(2)
set of semaphores, get	semget(2)
set or clear auditing on calling process	setaudproc(2)
set or display audit file information	audsys(1M)
set or get audit files	audctl(2)
set or get background character and rendition using a complex character	bkgrnd(3X)
set or get background character and rendition using a single-byte character	bkgd(3X)
set or get the thread process-shared attribute	pthread_condattr_getpshared(3T)
set or get tty baud rate	cfspeed(3C)
set printing options for a non-serial printer	slp(1)
set process group ID for job control	setpgid(2)
set process group ID	setpgrp(2)
set process group ID, create session and	setsid(2)
set process priority	getpriority(2)
set protocol entry	getprotoent(3N)
set real and effective user IDs	setreuid(2)
set real, effective, and/or saved user or group IDs	setresuid(2)
set scheduling parameters	rtsched(2)
set scheduling policy	rtsched(2)
set service entry	getservernt(3N)
set - set/define flags and arguments	csh(1)
set - set/define options and arguments	ksh(1)
set - set/define options and arguments	sh-bourne(1)
set - set/define options and arguments	sh-posix(1)
set signal alternate stack context	sigaltstack(2)
set special attributes for group	getprivgrp(2)
set system name	uname(1)
set system name	uname(2)
set system resource consumption limit	getrlimit(2)
set tabs on a terminal	tabs(1)
set terminal characteristics for cue	cuegetty(1M)
set terminal type, modes, speed and line discipline for 2-way line	uugetty(1M)
set terminal type, modes, speed, and line discipline	getty(1M)
set the blocking status of a message queue associated with a descriptor	mq_setattr(2)
set the cursor mode	curs_set(3X)
set the default message catalog	setcat(3)
set the interval timer	ualarm(2)
set the locale of a program	setlocale(3C)
set the prioceiling attribute	pthread_mutexattr_getprotocol(3T)
set the prioceiling of a mutex	pthread_mutex_getprioceiling(3T)
set the process-shared attribute	pthread_mutexattr_getpshared(3T)
set the process-shared attribute	pthread_rwlockattr_getpshared(3T)
set the protocol attribute	pthread_mutexattr_getprotocol(3T)
set the spin attribute	pthread_mutexattr_getspin_np(3T)
set the type attribute	pthread_mutexattr_getpshared(3T)
set the yield frequency attribute	pthread_mutexattr_getspin_np(3T)
set time and date	stime(2)
set time limit for I/O operations	io_timeout_ctl(3I)
set tty device operating parameters	tcattribute(3C)
set up I/O read termination character on special file	io_eol_ctl(3I)
set up NIS+ servers	nissserver(1M)
set user or group ID	setuid(2)
set value of process interval timer	getitimer(2)
set value of system-wide clock	setclock(3C)
set width (in bits) of data path	io_width_ctl(3I)
set, query configuration and loadable flags for a module	kmsystem(1M)
set, reset, or query system parameter	kmtune(1M)
set-group-ID bit	glossary(9)
set-user-ID bit	glossary(9)
set: file creation (permissions) mask, set and get	umask(2)

Description

Entry Name(Section)

set: file size limits and break value, get or set	ulimit(2)
setacl(), fsetacl() - set access control list (ACL) information	setacl(2)
setaclentry() - add, modify, or delete access control list entry	setaclentry(3C)
setaudit(2) - HP-UX Auditing System described	audit(5)
setaudit() - set audit ID (aid()) for current process	setaudit(2)
setaudproc() - set or clear auditing on calling process	setaudproc(2)
setboot - display and modify variables in the stable storage	setboot(1M)
setbuf(), setvbuf() - assign buffering to a stream file	setbuf(3S)
setcat() - set the default message catalog	setcat(3)
setcchar() - set cchar_t from a wide character string and rendition	setcchar(3X)
setclock() - set value of system-wide clock	setclock(3C)
setcontext() - get and set current user context	getcontext(2)
setdomainname() - set name of current NIS domain	getdomainname(2)
setdvagent() - set device assignment database entry	getdvagent(3)
setenv - define environment variable	csh(1)
setevent(2) - HP-UX Auditing System described	audit(5)
setevent() - set current events and system calls to be audited	setevent(2)
setexportent() - access exported file system information	exportent(3N)
setext - set extent attributes (VxFS)	setext(1M)
setfsent() - open and rewind file system descriptor file	getfsent(3X)
setgid() - set group ID	setuid(2)
setgrent() - rewind pointer to first entry in group() file	getgrent(3C)
setgroups() - set group access list	setgroups(2)
sethostent() - set network host entry	gethostent(3N)
sethostent_r() - set network host entry (thread-safe)	gethostent(3N)
sethostname() - set name of host cpu	sethostname(2)
setitimer() - set value of process interval timer	getitimer(2)
setjmp() - save stack environment for non-local goto	setjmp(3C)
setkey() - generate hashing encryption	crypt(3C)
setkey_r() - generate hashing encryption	crypt(3C)
setlabel() - define label for formatting routines	setlabel(3)
setlocale() - set the locale of a program	setlocale(3C)
setlocale_r() - set the locale of a program (MT-Safe)	setlocale(3C)
setlogmask() - set system log file priority mask	syslog(3C)
setmnt - establish mount table /etc/mnttab	setmnt(1M)
setmntent() - open a file system description file	getmntent(3X)
setnetconfig() - get /etc/netconfig entry corresponding to NETPATH component	getnetpath(3N)
setnetconfig() - get network configuration data base entry	getnetconfig(3N)
setnetent(): get network entry	getnetent(3N)
setnetgrent() - get network group entry	getnetgrent(3C)
setpgid() - set process group ID for job control	setpgid(2)
setpgrp() - create session and set process group ID	setsid(2)
setpgrp() - set process group ID	setpgrp(2)
setpgrp2() - set process group ID for job control	setpgid(2)
setpgrp3() - create session and set process group ID	setsid(2)
setprdfent() - manipulate system default database entry	getprdfent(3)
setpriority() - set process priority	getpriority(2)
setprivgrp - set special privileges for group	setprivgrp(1M)
setprivgrp() - set special attributes for group	getprivgrp(2)
setprotoent() - set protocol entry	getprotoent(3N)
setprotoent_r() - set protocol entry (thread-safe)	getprotoent(3N)
setprpwent() - set protected password database entry	getprpwent(3)
setprtcent() - manipulate terminal control database entry	getprtcent(3)
setpwent() - rewind pointer to beginning of password file	getpwent(3C)
setresgid() - set real, effective, and/or saved group IDs	setresuid(2)
setresuid() - set real, effective, and/or saved user IDs	setresuid(2)
setrlimit() - set system resource consumption limit	getrlimit(2)
setrlimit64() - file system API to support large files	creat64(2)
setscreg() - terminal output control functions	clearok(3X)
setservent() - set service entry	getservent(3N)
setservent_r() - set service entry (thread-safe)	getservent(3N)

Description	Entry Name(Section)
setsid() - create session and set process group ID	setsid(2)
setsockopt() - set options on sockets	getsockopt(2)
setspent() - rewind pointer to beginning of secure password file	getspent(3C)
setspwent() - rewind pointer to beginning of secure password file	getspwent(3X)
setstate(), random(), srandom(), initstate() - generate a pseudorandom number	random(3M)
settings, terminal, and datacomm line speed used by getty	gettydefs(4)
setuid() - set user ID	setuid(2)
setuname - change machine information	setuname(1M)
setuname() - set node name (system name)	uname(2)
setup, audio	see audio(5)
setupterm() - interface to terminfo database	del_curterm(3X)
setusershell() - rewind legal user shells file	getusershell(3C)
setutent() - reset input stream to beginning of utmp() file	getut(3C)
setutent_r() - reset input stream to beginning of utmp() file	getut(3C)
setutxent() - reset input stream to beginning of utmpx() file	getutx(3C)
set_curterm() - interface to terminfo database	del_curterm(3X)
set_parms special initialization script	hostname(1)
set_term() - switch between screens	set_term(3X)
severities, define additional	adddev(3C)
sh - overview of various system shells	sh(1)
sh - shell based on POSIX.2	sh-posix(1)
sh - shell, the standard command programming language	sh-bourne(1)
shar - make a shell archive package	shar(1)
shared cache file, NIS+ utility to print out the contents of	nisshowcache(1M)
shared libraries programs, prepare for faster program start-up	fastbind(1)
shared libraries, generate from object files	ld(1)
shared library	glossary(9)
shared library loaded module	dlget(3C)
shared library loaded module	dlmodinfo(3C)
shared library, list dynamic dependencies of shared libraries	ldd(1)
shared library, load or unload	shl_load(3X)
shared library, lock in memory	plock(2)
shared library, look up symbol in	shl_load(3X)
shared memory and data segment, attach or detach	shmop(2)
shared memory control operations	shmctl(2)
shared memory identifier (shmid)	glossary(9)
shared memory identifier, remove	ipcrm(1)
shared memory operation permissions	glossary(9)
shared memory segment, get	shmget(2)
shared memory segments, report status	ipcs(1)
shared object, close	dlclose(3C)
shared object, get address of symbol	dlsym(3C)
shared object, open	dlopen(3C)
shared strings, extract strings from C programs to implement	xstr(1)
shear characters from beginning of line and move to end of line	newform(1)
shell (command interpreter) with C-like syntax	csh(1)
shell archive package, make a	shar(1)
shell command, issue a	system(3S)
SHELL environment variable	login(1)
shell	glossary(9)
shell layer manager	shl(1)
shell procedures for system accounting	acctsh(1M)
shell program	glossary(9)
shell script	glossary(9)
shell script to set up user's environment at login	profile(4)
shell server, remote	remshd(1M)
shell, Bourne	sh-bourne(1)
shell, context-sensitive softkey	keysh(1)
shell, login, change default	chsh(1)
shell, remote, execute from a	remsh(1)
shell, restricted shell for sendmail	smrsh(1M)

Index

All Volumes

Description	Entry Name(Section)
shells - list of allowed login shells	shells(4)
shells, get legal user	getusershell(3C)
shells, list of allowed login	shells(4)
shells, overview of various system	sh(1)
shift - shift <i>argv</i> members one position to left	csh(1)
shift - shift <i>argv</i> members one position to left	ksh(1)
shift - shift <i>argv</i> members one position to left	sh-posix(1)
shift - shift positional parameters to next lower position	sh-bourne(1)
shl - shell layer manager	shl(1)
shl_findsym() - look up symbol in shared library	shl_load(3X)
shl_get() - get information about shared library	shl_load(3X)
shl_load() - load shared library	shl_load(3X)
shl_unload() - unload shared library	shl_load(3X)
shmat() - attach shared memory to data segment	shmop(2)
shmctl() - shared memory control operations	shmctl(2)
shmdt() - detach shared memory from data segment	shmop(2)
shmget() - get shared memory segment	shmget(2)
shmid (shared memory identifier)	glossary(9)
shm_open() - create/open a shared memory object	shm_open(2)
shm_unlink() - unlink a shared memory object	shm_unlink(2)
shortest path and route between hosts, compute	pathalias(1)
show all remote mounts	showmount(1M)
show current system-clock date and time	date(1)
show disk usage	du(1)
show FDDI interface status	fddistat(1M)
show Fibre Channel Light Weight Protocol network status	lwpstat(1M)
show file differences side-by-side	sdiff(1)
show group memberships	groups(1)
show how long system has been up	uptime(1)
show network status	netstat(1)
show PCI FDDI interface status	fddipciadmin(1M)
show STREAMS structures	strdb(1M)
show who is logged in on local machines	rwho(1)
showmount - show all remote mounts	showmount(1M)
shut down a socket	shutdown(2)
shut down and reboot the system	reboot(1M)
shutacct - turn off accounting for system shutdown	acctsh(1M)
shutdown status of specified file system, determine	fsclean(1M)
shutdown - terminate all processing	shutdown(1M)
shutdown() - shut down a socket	shutdown(2)
shutdown, system, turn accounting off for	acctsh(1M)
shutdown, test for file system clean at last	fsclean(1M)
sigaction() - examine and change signal action	sigaction(2)
sigaddset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigaltstack() - set and/or get signal alternate stack context	sigaltstack(2)
sigblock() - block signals	sigblock(2)
sigdelset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigemptyset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sigfillset() - initialize, manipulate, and test signal sets	sigsetops(3C)
sighold(), sigrelse(), sigignore() - signal management	sighold(2V)
sighold() - signal management	signal(2)
sigignore(), sighold(), sigrelse() - signal management	sighold(2V)
sigignore() - signal management	signal(2)
siginterrupt() - allow signals to interrupt functions	siginterrupt(2)
sigismember() - initialize, manipulate, and test signal sets	sigsetops(3C)
sign on	login(1)
sign-determination macro, floating-point	signbit(3M)
signal action, examine and change	sigaction(2)
signal action, examine and change	sigwait(2)
signal alternate stack context, set and/or get	sigaltstack(2)
signal - description of signals	signal(5)

Description	Entry Name(Section)
signal facilities	bsd_signal(3C)
signal facilities, software	sigvector(2)
signal	glossary(9)
signal management (sighold() , sigrelse() , sigignore())	sighold(2V)
signal management (sigset() , sighold() , sigrelse() , sigignore() , sigpause())	sigset(3C)
signal mask, examine and change, of the calling thread	pthread_sigmask(3T)
signal sets, initialize, manipulate, and test	sigsetops(3C)
signal stack context, set and/or get	sigstack(2)
signal stack space, define, delete, or get amount of	sigspace(2)
signal to a thread, send	pthread_kill(3T)
signal to process	kill(1)
signal() - 4.2 BSD-compatible signal() system call	killpg(2)
signal() - specify what to do upon receipt of a signal	signal(2)
signal() system call, 4.2 BSD-compatible	bsdproc(3C)
signal() system call, 4.2 BSD-compatible	killpg(2)
signal, audible	beep(3X)
signal, define what to do upon receipt of a	signal(2)
signal, hold upon receipt	sighold(2V)
signal, hold upon receipt	sigset(3C)
signal, ignore	sighold(2V)
signal, ignore	sigset(3C)
signal, raise a software	ssignal(3C)
signal, restore action	sighold(2V)
signal, restore action	sigset(3C)
signal, select method of handling	sighold(2V)
signal, select method of handling	sigset(3C)
signal, send to a process or a group of processes	sigsend(2)
signal, send to process or group of processes	kill(2)
signal, suspend calling process until received	sighold(2V)
signal, suspend calling process until received	sigset(3C)
signal, suspend process until	pause(2)
signal, wait for a	sigsuspend(2)
signal.h - description of signals	signal(5)
signals allowed to interrupt functions	siginterrupt(2)
signals mask, set current ignorable	sigsetmask(2)
signals, block	sigblock(2)
signals, blocked, examine and change	sigprocmask(2)
signals, description of	signal(5)
signals, examine pending	sigpending(2)
signals, release blocked and atomically wait for interrupt	sigpause(2)
signals, send to domain name server	sig_named(1M)
signbit() - floating-point sign-determination macro	signbit(3M)
signgam() , gamma() , lgamma() , lgamma_r() - log gamma function	lgamma(3M)
sigpause() - atomically release blocked signals and wait for interrupt	sigpause(2)
sigpending() - examine pending signals	sigpending(2)
sigprocmask() - examine and change blocked signals	sigprocmask(2)
sigqueue() - queue a signal to a process	sigqueue(2)
sigrelse() , sigignore() , sighold() - signal management	sighold(2V)
sigrelse() - signal management	signal(2)
sigsend() - send a signal to a process	sigsend(2)
sigsendset() - send a signal to a group of processes	sigsend(2)
sigset() , sighold() , sigrelse() , sigignore() , sigpause() - signal management	sigset(3C)
sigset() - signal management	signal(2)
sigsetjmp() - save signal mask if savemask is non-zero	setjmp(3C)
sigsetmask() - set current signal mask	sigsetmask(2)
sigspace() - define or delete additional signal stack space	sigspace(2)
sigstack() - set and/or get signal stack context	sigstack(2)
sigsuspend() - wait for a signal	sigsuspend(2)
sigtimedwait() - synchronously accept a signal	sigwait(2)
sigvec() - 4.2 BSD-compatible sigvec() system call	killpg(2)
sigvec() system call, 4.2 BSD-compatible	bsdproc(3C)

Index

All Volumes

Description	Entry Name(Section)
sigvec() system call, 4.2 BSD-compatible	killpg(2)
sigvector() - software signal facilities	sigvector(2)
sigwait() - synchronously accept a signal	sigwait(2)
sigwaitinfo() - synchronously accept a signal	sigwait(2)
sig_named - send signals to domain name server	sig_named(1M)
simple command, execute a	command(1)
simple text formatter	fmt(1)
simplified signal facilities	bsd_signal(3C)
sin() - sine function	sin(3M)
sind() - sine function (degrees)	sind(3M)
sindf() - sine function (float, degrees)	sind(3M)
sine function (degrees)	sind(3M)
sine function, inverse hyperbolic	asinh(3M)
sine functions	sin(3M)
sine functions, hyperbolic	sinh(3M)
sinf() - sine function (float)	sin(3M)
single-byte character and rendition, add, to a window and advance the cursor	addch(3X)
single-byte character and rendition, echo to a window and refresh	echochar(3X)
single-byte character and rendition, input from a window	inch(3X)
single-byte character and rendition, insert into a window	insch(3X)
single-byte character, get from the terminal	getch(3X)
single-byte character, set or get background character or rendition, using	bkgd(3X)
single-byte characters and renditions to a window, add length limited string of	addchnstr(3X)
single-byte characters and renditions to a window, add string of	addchstr(3X)
single-byte characters and renditions, array of, input from a window	inchnstr(3X)
single-byte characters and renditions, draw borders	border(3X)
single-byte characters and renditions, draw borders	box(3X)
single-byte characters and renditions, draw lines from	hline(3X)
single-byte terminal environment query functions	erasechar(3X)
single-user mode	init(1M)
single-user mode, place system in	shutdown(1M)
single-user state	glossary(9)
sinh() , sinhf() - hyperbolic sine functions	sinh(3M)
sinhf() , sinh() - hyperbolic sine functions	sinh(3M)
sis - secure internet services description	sis(5)
sixteen-bit characters, tools to process	nl_tools_16(3X)
size information of screen, specify source	use_env(3X)
size of file in words, lines, and bytes or characters	wc(1)
size - print section sizes and allocation space of object files	size(1)
size, extend file system	extendfs(1M)
size, extend HFS file system	extendfs_hfs(1M)
size, extend VxFS file system	extendfs_vxfs(1M)
size, return for elf32 or elf64 files	elf_fsize(3E)
slash	glossary(9)
slave (pseudo-terminal) driver, STREAMS	tels(7)
slave and master pty, STREAMS, unlocking	unlockpt(3C)
slave pty (pseudo-terminal) driver, STREAMS	pts(7)
slave pty, get the name of a	ptsname(3C)
slave pty, STREAMS, granting access	grantpt(3C)
sleep - suspend execution for a time interval	sleep(1)
sleep() - suspend execution for interval	sleep(3C)
sleep, high resolution	nanosleep(2)
slk_attoff() - soft label functions	slk_attoff(3X)
slk_attron() - soft label functions	slk_attoff(3X)
slk_attrset() - soft label functions	slk_attoff(3X)
slk_attr_off() - soft label functions	slk_attoff(3X)
slk_attr_on() - soft label functions	slk_attoff(3X)
slk_attr_set() - soft label functions	slk_attoff(3X)
slk_clear() - soft label functions	slk_attoff(3X)
slk_color() - soft label functions	slk_attoff(3X)
slk_init() - soft label functions	slk_attoff(3X)

Description	Entry Name(Section)
slk_label() - soft label functions	slk_attroff(3X)
slk_noutrefresh() - soft label functions	slk_attroff(3X)
slk_refresh() - soft label functions	slk_attroff(3X)
slk_restore() - soft label functions	slk_attroff(3X)
slk_set() - soft label functions	slk_attroff(3X)
slk_touch() - soft label functions	slk_attroff(3X)
slk_wset() - soft label functions	slk_attroff(3X)
slot in the utmp() file of the current user, find	ttyslot(3C)
slp - set printing options for a non-serial printer	slp(1)
sm - statd directory and file structures	sm(4)
sm.bak - statd directory and file structures	sm(4)
Small Computer System Interface (SCSI) device drivers	scsi(7)
Smart Card Login	login(1)
smrsh - restricted shell for sendmail	smrsh(1M)
sna3179g , sna3270 , sna3770 - IBM 3179G/3192G, 3270, 3777 terminal emulator	sna(1)
snapshot of the UUCP system	uusnap(1M)
SNMP agent, configuration file for the	snmpd.conf(4)
SNMP requests, daemon that responds to	snmpd(1M)
snmpd - daemon that responds to SNMP requests	snmpd(1M)
snmpd.conf - configuration file for the SNMP agent	snmpd.conf(4)
snprintf() - print formatted output to a string	printf(3S)
socket address, get	getsockname(2)
socket - Interprocess communications	socket(7)
socket() - create an endpoint for communication	socket(2)
socket, accept connection on a	accept(2)
socket, bind address to a	bind(2)
socket, bind to a privileged IP port	bindresvport(3N)
socket, initiate connection	connect(2)
socket, listen for connections on a	listen(2)
socket, receive message from a	recv(2)
socket, return a reserved port	rcmd(3N)
socket, send message to a	send(2)
socket, send the contents of a file through a socket	sendfile(2)
socket, shut down a	shutdown(2)
socketpair() - create a pair of connected sockets	socketpair(2)
sockets, create a pair of connected	socketpair(2)
sockets, get or set options	getsockopt(2)
soelim - eliminate .so's from ncroff input	soelim(1)
soft (symbolic) file link	symlink(4)
soft label functions	slk_attroff(3X)
soft-copy terminals, peruse file on	more(1)
soft-copy terminals, peruse file on	pg(1)
softbench - SoftBench Software Development Environment	softbench(1)
SoftBench Software Development Environment	softbench(1)
softkey file format for keysh	softkeys(4)
softkey shell, context-sensitive	keysh(1)
softkeys - keysh softkey file format	softkeys(4)
softpower - determine if softpower hardware is installed	softpower(1M)
softpower hardware	softpower(1M)
Software Development Environment, SoftBench	softbench(1)
Software Distributor agent on remote systems	pushAgent(1M)
software products, display information about	swlist(1M)
software products, install, configure, and copy	swinstall(1M)
software products, modify in target root or depot	swmodify(1M)
software products, package into target depot or tape	swpackage(1M)
software products, remove and unconfigure	swremove(1M)
software products, verify	swverify(1M)
software signal facilities	sigvector(2)
software signal, raise a	ssignal(3C)
software signal, send to process or group of processes	kill(2)
software, outbound connection daemon debug utility used by DDFA	ocdebug(1M)

Index

All Volumes

Description	Entry Name(Section)
solidus	glossary(9)
sort a directory pointer array	scandir(3C)
sort and embellish uusnap output	uusnaps(1M)
sort and/or merge files	sort(1)
sort - sort and/or merge files	sort(1)
sort, quicker	qsort(3C)
sort, topological	tsort(1)
sorted files, reject/select lines common to two	comm(1)
sorted tables, binary search routine for	bsearch(3C)
Source Code Control System (SCCS)	glossary(9)
Source Code Control System	see SCCS
source code	glossary(9)
source - define source for command input	csh(1)
source of screen size information, specify	use_env(3X)
source program files for given name, find location of	whereis(1)
source, C, extract error messages from into a file	mkstr(1)
space allocation, change data segment	brk(2)
space for LVM logical volume, increase	lvextend(1M)
space for signal stack, define, delete, or get amount of	sigspace(2)
space information, system paging	swapinfo(1M)
space, recover disk	freedisk(1M)
space, stack and data, allocate then lock process into memory	datalock(3C)
spaces, convert to tabs and vice versa	expand(1)
spawn getty to remote terminal (call terminal)	ct(1)
spawn new process (use fork() instead)	vfork(2)
spawn processes	init(1M)
spd - set physical drive parameters for an HP SCSI disk array	spd(1M)
special (device) file, make	mksf(1M)
special (device) file, remove a	rmsf(1M)
special (device) files, install	insf(1M)
special (I/O device) file, list a	lssf(1M)
special and FIFO files, create	mknod(1M)
special attributes for group, get	getprivgrp(1)
special file	glossary(9)
special file, control character device	ioctl(2)
special file, FIFO, make a	mkfifo(3C)
special file, make	mknod(2)
special file, set up I/O read termination character on	io_eol_ctl(3I)
special files, introduction	intro(7)
special files, make FIFO (named pipe)	mkfifo(1)
special files: cartridge tape device	ct(7)
special files: general terminal interface	termio(7)
special files: HP-HIL cooked keyboard driver	hilkbd(7)
special files: HP-HIL device driver	hil(7)
special files: line printer device files	lp(7)
special files: pseudo-terminal driver	pty(7)
special files: system console interface	console(7)
special files: terminal interface device file, process-controlling	tty(7)
special files: Version 6/PWB-compatible terminal interface	sttyv6(7)
special files; list all device drivers available in the system	lsdev(1M)
special functions of HP 2640- and HP 2621-series terminals, handle	hp(1)
special initialization script, /sbin/set_parms	hostname(1)
special processes	glossary(9)
specific Network Information Service server, bind to a	ypset(1M)
specific time interval to the current absolute system time, add a	get_expiration_time(3T)
specification, format, in text files	fspec(4)
specified remote machines, write to	rwall(3N)
specify I/O read termination character on special file	io_eol_ctl(3I)
specify source of screen size information	use_env(3X)
specify what to do upon receipt of a signal	signal(2)
speed, datacomm line, and terminal settings used by getty	gettydefs(4)

Description	Entry Name(Section)
speed, inform system of required minimum I/O transfer	io_speed_ctl(3I)
spell - find spelling errors	spell(1)
spellin - convert 9-digit hash codes to compressed spelling reference list	spell(1)
spelling errors, find	spell(1)
spin attribute, get or set	pthread_mutexattr_getspin_np(3T)
split a file into multiple <i>n</i> -line pieces	split(1)
split buffer into fields	bufsplit(3G)
split file into multiple files	csplit(1)
split mirrored LVM logical volume into two logical volumes	lvsplit(1M)
split - split a file into multiple <i>n</i> -line pieces	split(1)
spool directory clean-up, uucp	uuclean(1M)
spool directory clean-up, uucp	uucleanup(1M)
spooled uucp transactions grouped by transaction, list	uuls(1M)
spooler performance analysis information, print LP	lpana(1M)
spooling	see printer
spooling system, configure the LP	lpadmin(1M)
spooling system, line printer	see LP
spray : scatter data to check the network	spray(3N)
spray packets	spray(1M)
spray server	sprayd(1M)
spray - spray packets	spray(1M)
sprayd - spray server	sprayd(1M)
sprintf() - print formatted output to a string	printf(3S)
sqrt() sqrtf() - square root functions	sqrt(3M)
sqrtf() , sqrt() - square root functions	sqrt(3M)
square root functions	sqrt(3M)
srand() - reset random-number generator to random starting point	rand(3C)
srand48() , seed48() , long48() - initialize pseudo-random number generator	drand48(3C)
srandom() , initstate() , setstate() , random() - generate a pseudorandom number	random(3M)
SRQ line on HP-IB, allow interface to enable	hpib_rqst_srvce(3I)
SS/80	glossary(9)
sscanf() - formatted read from character string	scanf(3S)
ssignal() - raise a software signal and perform an action	ssignal(3C)
ssp - remove multiple line-feeds from output	ssp(1)
ssrfc - SCSI surface device access	autochanger(7)
sss - set spindle synchronization state of drives in an HP SCSI disk array	sss(1M)
stable storage, display and modify variables in	setboot(1M)
stack and data space, allocate then lock process into memory	datalock(3C)
stack context, set/get signal alternate stack context	sigaltstack(2)
stack context, signal, set and/or get	sigstack(2)
stack environment, save/restore for non-local goto	setjmp(3C)
stack space for signals, define, delete, or get amount of	sigspace(2)
stack, lock in memory	plock(2)
stacksize, change default	pthread_default_stacksize_np(3T)
stale logical volume mirrors in LVM volume groups, synchronize	vgsync(1M)
stale mirrors in LVM logical volumes, synchronize	lvsync(1M)
standard buffered input/output stream file package	stdio(3S)
standard error and console, displays formatted message on	fmtmsg(3C)
standard error	glossary(9)
standard format, display message in	pfmt(3C)
standard input	glossary(9)
standard input stream, input string from a	gets(3S)
standard input to system log, send	logger(1)
standard input, read a line from	read(1)
standard output	glossary(9)
standard output, send copy of to specified file	tee(1)
standard structures and symbolic constants	unistd(5)
standend() - set and clear window attributes	standend(3X)
standout() - set and clear window attributes	standend(3X)
stape, magnetic tape interface for tape1, tape2, and	mt(7)
start accounting process at system startup	acctsh(1M)

Index

All Volumes

Description	Entry Name(Section)
start license server	i4lmd(1M)
start of file, list first few lines at	head(1)
start or halt auditing system	audctl(2)
start or halt auditing system	audsys(1M)
start terminal session	login(1)
start the Virtual Home Environment (VHE)	vhe_mounter(1M)
start/stop the LP spooling request scheduler	lpsched(1M)
startup - start accounting process at system startup	acctsh(1M)
startup, system, start accounting process at	acctsh(1M)
start_color() - color manipulation functions	can_change_color(3X)
stat/fstat/lstat system call, data returned by	stat(5)
stat - data returned by stat/fstat/lstat system call	stat(5)
stat() - get file status	stat(2)
stat64() - file system API to support large files	creat64(2)
statd directory and file structures	sm(4)
statd - network status monitor	statd(1M)
state information, get Terminal Session Manager	tsm.info(1)
state - statd directory and file structures	sm(4)
state with its state on disk, synchronize a file's in-core	fsync(2)
state, PAM routines to maintain module specific state	pam_set_data(3)
statfs(), fstatfs() - get file system statistics	statfs(2)
statfsdev(), fstatfsdev() - get file system statistics	statfsdev(3C)
static file system mounting table	pfs_fstab(5)
static information about the file systems	fstab(4)
station address string conversion routines, network	net_aton(3C)
statistics for file system, list (generic)	ff(1M)
statistics for HFS file system, list file names and	ff_hfs(1M)
statistics for VxFS file system	ff_vxfs(1M)
statistics server, kernel	rstatd(1M)
statistics, get file system	statfs(2)
statistics, get file system	statfsdev(3C)
statistics, get file system	statvfsdev(3C)
statistics, get mounted file system	ustat(2)
statistics, local file system mount	rmtab(4)
statistics, Network File System	nfsstat(1M)
statistics, print mail traffic	mailstats(1)
statistics, report I/O	iostat(1)
statistics, report NIS+ server	nisstat(1M)
statistics, report virtual memory	vmstat(1)
status condition becomes true, wait until the requested	hpib_status_wait(3I)
status information and attributes associated with a message queue, get	mq_getattr(2)
status information, LP request, print	lpstat(1)
status inquiries, stream	ferror(3S)
status inquiry and job control, uucp	uustat(1)
status lines of GPIO card, return	gpio_get_status(3I)
status monitor, network	statd(1M)
status of HP-IB interface, return	hpib_bus_status(3I)
status of interprocess communication facilities, report	ipcs(1)
status of local machines, show	ruptime(1)
status server, system	rwhod(1M)
status show network	netstat(1)
status, asynchronous I/O error	aio_error(2)
status, audit, of event or system call, change or display	audevent(1M)
status, current, of the UUCP system	uusnap(1M)
status, display LAN device configuration and	lanscan(1M)
status, exit, do nothing and return zero or non-zero	true(1)
status, get file	stat(2)
status, get file status	fstat(2)
status, get file	statvfs(2)
status, get symbolic link	lstat(2)
status, host, of local machines (RPC version), show	rup(1)

Description	Entry Name(Section)
status, line update status functions	redrawwin(3X)
status, report process	ps(1)
status, show FDDI interface	fddistat(1M)
status, show Fibre Channel Light Weight Protocol network	lwpstat(1M)
status, show PCI FDDI interface	fddipciadmin(1M)
status: LP requests sent to remote system for printing	rlpstat(1M)
statvfs() , fstatvfs() - get file status	statvfs(2)
statvfs64() - file system API to support large files	creat64(2)
statvfsdev() , fstatvfsdev() - get file system statistics	statvfsdev(3C)
statvfsdev64() - file system API to support large files	fgetpos64(2)
stdarg.h - macros for handling variable argument list	stdarg(5)
stderr	glossary(9)
stdin	glossary(9)
stdio() - standard buffered input/output stream file package	stdio(3S)
stdout	glossary(9)
stdscr() - default window	stdscr(3X)
stdsyms - description of HP-UX header file organization	stdsyms(5)
step() - regular expression string comparison routine	regex(3X)
sticky bit	glossary(9)
stime() - set time and date	stime(2)
stop activity on specified HP-IB	hplib_abort(3I)
stop and reset FDDI interface	fddistop(1M)
stop servers	pdshutdown(1)
stop system operation	shutdown(1M)
stop then reboot the system	reboot(1M)
stop, wait for child process to	wait(2)
stop/start the LP spooling request scheduler	lpsched(1M)
stops printers from accepting jobs and logs from logging	pddisable(1)
storage associated with a screen	delscreen(3X)
storage formats, attributes, and objects for SD	sd(4)
storage, preallocate disk	prealloc(1)
storage, preallocate fast disk	prealloc(2)
storage, stable, display and modify variables in	setboot(1M)
store() - store data under a key (old single-data-base version)	dbm(3X)
strace - write STREAMS event trace messages to standard output	strace(1M)
strcasecmp() , strncasecmp() - compare two strings	string(3C)
strcat() , strncat() - append string 2 to string 1	string(3C)
strchg - change or query stream configuration	strchg(1M)
strchr() , strrchr() - get pointer to character in string	string(3C)
strclean - remove outdated STREAMS error log files	strclean(1M)
strcmp() , strncmp() - compare two strings	string(3C)
strcoll() - process string of text tokens	string(3C)
strconf - query stream configuration	strchg(1M)
strcpy() , strncpy() - copy string 2 to string 1	string(3C)
strcsn() , strspn() - find length of matching substrings	string(3C)
strdb - STREAMS debugging tool	strdb(1M)
strdup() - determine length of a string	string(3C)
stream configuration, change or query	strchg(1M)
stream creation, library routines for external data representation	xdr_create(3N)
stream file or character string, read from with formatted input conversion	scanf(3S)
stream file package, standard buffered input/output	stdio(3S)
stream file, assign buffering to a	setbuf(3S)
stream file, buffered binary input/output to a	fread(3S)
stream file, get character or data word from a	getc(3S)
stream file, get or reposition pointer for I/O operations on a	fseek(3S)
stream file, get wide character from a	getwc(3C)
stream file, input wide string from a	fgetws(3C)
stream file, open or re-open; convert file to stream	fopen(3S)
stream	glossary(9)
stream pointer, map to file descriptor	fileno(3S)
stream status inquiries	error(3S)

Index

All Volumes

Description	Entry Name(Section)
stream() - STREAMS enhancements to standard system calls	stream(2)
stream, close a	fclose(3S)
stream, flush buffer with or without closing	fclose(3S)
stream, input string from a standard input	gets(3S)
stream, push character back into input	ungetc(3S)
stream, push wide character back into input	ungetwc(3C)
stream, put wide character on a	putwc(3C)
stream, put word or character on a	putc(3S)
stream, read up to next delimiter	bgets(3G)
stream, return to a remote command	rcmd(3N)
stream, return to a remote command	rexec(3N)
stream, save or restore file position indicator for a	fgetpos(3S)
streaming text editor	sed(1)
streamio - STREAMS ioctl commands	streamio(7)
STREAMS Administrative Driver	sad(7)
STREAMS debugging tool	strdb(1M)
STREAMS enhancements to system calls	stream(2)
STREAMS error log files, remove outdated files	strclean(1M)
STREAMS event trace messages to standard output, write	strace(1M)
STREAMS file descriptor	fattach(3C)
STREAMS file, receive next message	getmsg(2)
STREAMS ioctl commands	streamio(7)
STREAMS log driver	strlog(7)
STREAMS log driver, receive error messages	strerr(1M)
STREAMS master pty (pseudo-terminal) driver	ptm(7)
STREAMS module for converting ioctl() calls into Transport Interface messages	timod(7)
STREAMS module for reads and writes by Transport Interface	tirdwr(7)
STREAMS module, terminal line discipline	ldterm(7)
STREAMS modules, manage system database of automatically pushed STREAMS modules	autopush(1M)
STREAMS pass through device driver to open a major and minor device pair on a STREAMS driver	clone(7)
STREAMS pty (pseudo-terminal) Emulation module	ptem(7)
STREAMS pty master/slave pair, unlocking	unlockpt(3C)
STREAMS pty, get the name of a slave	ptsname(3C)
STREAMS pty, Packet Mode module	pckt(7)
STREAMS slave pty driver	pts(7)
STREAMS slave pty, granting access to	grantpt(3C)
STREAMS structures show	strdb(1M)
STREAMS system calls	stream(2)
STREAMS Telnet slave driver	tels(7)
STREAMS terminal line discipline module	ldterm(7)
STREAMS verification tool	strvf(1M)
streams within a multi-thread application, explicit locking of	flockfile(3S)
STREAMS, change or query stream configuration	strchg(1M)
streams, directory, format of	dirent(5)
streams, HP-UX directory, format of	ndir(5)
STREAMS, send a message on a stream	putmsg(2)
STREAMS-based file descriptor	fdetach(1M)
STREAMS-based file descriptor	fdetach(3C)
STREAMS-based pipe	isastream(3C)
STREAMS: attach a STREAMS file descriptor	fattach(3C)
STREAMS: detach a name from a STREAMS-based file descriptor	fdetach(3C)
STREAMS: detach a STREAMS-based file descriptor	fdetach(1M)
STREAMS: determine if file descriptor refers to STREAMS device or STREAMS-based pipe	isastream(3C)
strerr - receive error messages from the STREAMS log driver	strerr(1M)
strerror() - system error messages	perror(3C)
strfmon() - convert monetary value to string	strfmon(3C)
strftime() - convert date and time to string	strftime(3C)
string conversion routines, network station address	net_aton(3C)
string data order, convert	strord(3C)
string form, convert access control list (ACL) structure to	acltostr(3C)
string from a standard input stream, input	gets(3S)

Description	Entry Name(Section)
string of single-byte characters and renditions to a window, add	addchstr(3X)
string of wide characters, input from a window	innwstr(3X)
string of wide-character, insert into a window	ins_nwstr(3X)
string operations, character	string(3C)
string operations, wide character	wcstring(3C)
string or expression, search a file for a	grep(1)
string or string array element, convert floating-point number to	ecvt(3C)
string pointer for ELF files, make	elf_strptr(3E)
string to double-precision number, convert	strtod(3C)
string to long integer, convert	strtol(3C)
string, convert between long integer and base-64 ASCII	a64l(3C)
string, convert date and time to	ctime(3C)
string, convert date and time to	strftime(3C)
string, convert date and time to wide-character	wcsftime(3C)
string, convert long double floating-point number to	ldcv(3C)
string, convert long integer to	ltostr(3C)
string, convert monetary value to	strfmon(3C)
string, convert to access control list (ACL) structure	strtoacl(3C)
string, convert to long double-precision number	strtold(3C)
string, get a multi-byte character length limited string from the terminal	getnstr(3X)
string, get a multi-byte character string from the terminal	getstr(3X)
string, parse suboptions from a	getsubopt(3C)
string-to-NaN conversion function	nan(3M)
string-valued configuration values, get	confstr(3C)
strings and characters conversions, multibyte	multibyte(3C)
strings - find the printable strings in an object, or other binary, file	strings(1)
strings, concatenate two	string(3C)
strings, find for inclusion in message catalogs	findstr(1)
strings: extract strings from C programs to implement shared strings	xstr(1)
strip nroff/troff, tbl, and neqn constructs from a file	deroff(1)
strip - strip symbol and line number information from an object file	strip(1)
strip symbol and line number information from an object file	strip(1)
stripe LVM logical volume	lvextend(1M)
strlen() - determine length of a string	string(3C)
strlog - STREAMS log driver	strlog(7)
strord() - convert string data order	strord(3C)
strprkr() - find occurrence of character from string 2 in string 1	string(3C)
strptime() date and time conversion	strptime(3C)
strrstr() - process string of text tokens	string(3C)
strspn(), strcspn() - find length of matching substrings	string(3C)
strstr() - process string of text tokens	string(3C)
strtoacl() - convert string to access control list (ACL) structure	strtoacl(3C)
strtoaclpatt() - convert string to access control list (ACL) structure	strtoacl(3C)
strtod() - convert string to double-precision number	strtod(3C)
strtok() - process string of text tokens	string(3C)
strtok_r() - process string of text tokens	string(3C)
strtol() - convert string to long integer	strtol(3C)
strtold() - convert string to long double-precision number	strtold(3C)
strtoul() - convert string to long integer	strtol(3C)
structure format, symbol table	nlist(4)
structure, Network Information Service database and directory	ypfiles(4)
structures and symbolic constants, standard	unistd(5)
structures show STREAMS	strdb(1M)
structures, statd directory and file	sm(4)
strvf - STREAMS verification tool	strvf(1M)
strxfrm() - process string of text tokens	string(3C)
stty - set the options for a terminal port	stty(1)
stty - terminal interface for Version 6/PWB compatibility	sttyv6(7)
stty() - control terminal device (Bell Version 6 compatibility)	stty(2)
su - switch user	su(1)
subdirectories in directory, list	ls(1)

Index

All Volumes

Description	Entry Name(Section)
subdirectory	glossary(9)
submits print jobs	pdpr(1)
suboptions, parse from a string	getsubopt(3C)
subordinate directory	glossary(9)
subpad() - enhanced pad management function	subpad(3X)
subroutine call graph execution profile data, display	gprof(1)
subroutines and libraries, introduction	intro(3)
subroutines, database (new multiple database version)	ndbm(3X)
subroutines, database (old version - see also ndbm(3X))	dbm(3X)
subroutines, NIS+	nis_subr(3N)
Subset 1980	glossary(9)
substitute selected characters	tr(1)
subwin() - window creation functions	newwin(3X)
suffix - file-name suffix conventions	suffix(5)
sum - print checksum and block count of a file	sum(1)
summarize and print an SCCS file	prs(1)
summarize disk usage	du(1)
summarize file system ownership	quot(1M)
summarize file system ownership	quot_vxfs(1M)
summarize file system quotas	repquota(1M)
summarize, add, modify, delete, or copy file access control lists (ACLs)	chacl(1)
summary files, accounting, create periodic	acctsh(1M)
summary log of uucp and uux transactions, access	uucp(1)
super-block, update	sync(2)
superblock	glossary(9)
superior directory	glossary(9)
superuser	glossary(9)
superuser	passwd(1)
superuser, change login name to	su(1)
supplementary group ID	glossary(9)
support for local network packet routing, system	routing(7)
supported terminal video attributes, get	termattrs(3X)
suppress echo while reading password from terminal	getpass(3C)
surface device access module, SCSI	autochanger(7)
suspend Curses session	endwin(3X)
suspend execution for a time interval	sleep(1)
suspend execution for an interval	usleep(2)
suspend execution for interval	sleep(3C)
suspend execution of a thread	pthread_resume_np(3T)
suspend for asynchronous I/O completion	aio_suspend(2)
suspend foreground until background processes are finished	wait(1)
suspend or resume auditing on current process	audswitch(2)
suspend process until signal	pause(2)
suspend the calling process	napms(3X)
svcerr_auth() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_decode() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_noproc() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_noprogram() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_progvers() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_systemerr() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcerr_weakauth() - library routines for server side remote procedure call errors	rpc_svc_err(3N)
svcfld_create() - obsolete library routines for RPC	rpc_soc(3N)
svccraw_create() - obsolete library routines for RPC	rpc_soc(3N)
svctcp_create() - obsolete library routines for RPC	rpc_soc(3N)
svcdup_bufcreate() - obsolete library routines for RPC	rpc_soc(3N)
svcdup_create() - obsolete library routines for RPC	rpc_soc(3N)
svc_auth_reg() - library routines for registering servers	rpc_svc_reg(3N)
svc_control() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_create() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_destroy() - library routines for the creation of server handles	rpc_svc_create(3N)
svc_dg_create() - library routines for the creation of server handles	rpc_svc_create(3N)

Description	Entry Name(Section)
<code>svc_dg_enablecache()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_done()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_exit()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_fds()</code> - obsolete library routines for RPC	<code>rpc_soc(3N)</code>
<code>svc_fdset()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_fd_create()</code> - library routines for the creation of server handles	<code>rpc_svc_create(3N)</code>
<code>svc_freeargs()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_getargs()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_getcaller()</code> - obsolete library routines for RPC	<code>rpc_soc(3N)</code>
<code>svc_getreq()</code> - obsolete library routines for RPC	<code>rpc_soc(3N)</code>
<code>svc_getreqset()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_getreq_common()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_getreq_poll()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_getrpcaller()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_pollset()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_raw_create()</code> - library routines for the creation of server handles	<code>rpc_svc_create(3N)</code>
<code>svc_reg()</code> - library routines for registering servers	<code>rpc_svc_reg(3N)</code>
<code>svc_register()</code> - obsolete library routines for RPC	<code>rpc_soc(3N)</code>
<code>svc_run()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_sendreply()</code> - library routines for RPC servers	<code>rpc_svc_calls(3N)</code>
<code>svc_tli_create()</code> - library routines for the creation of server handles	<code>rpc_svc_create(3N)</code>
<code>svc_tp_create()</code> - library routines for the creation of server handles	<code>rpc_svc_create(3N)</code>
<code>svc_unreg()</code> - library routines for registering servers	<code>rpc_svc_reg(3N)</code>
<code>svc_unregister()</code> - obsolete library routines for RPC	<code>rpc_soc(3N)</code>
<code>svc_vc_create()</code> - library routines for the creation of server handles	<code>rpc_svc_create(3N)</code>
<code>swab()</code> - swap bytes	<code>swab(3C)</code>
<code>swacl</code> - view or modify Access Control Lists	<code>swacl(1M)</code>
<code>swagent</code> - perform software management tasks as the agent of an SD command	<code>swagentd(1M)</code>
<code>swagentd</code> - serve local or remote software management tasks	<code>swagentd(1M)</code>
<code>swap bytes</code>	<code>swab(3C)</code>
<code>swap device</code> for interleaved paging and swapping, add a	<code>swapon(2)</code>
<code>swap space information, system</code>	<code>swapinfo(1M)</code>
<code>swap</code> the left-to-right text character sequence in each line of a file	<code>rev(1)</code>
<code>swap</code> up to eight characters (to first tab in line) to end of line.....	<code>newform(1)</code>
<code>swap volume, prepare LVM logical volume to be</code>	<code>lvlnboot(1M)</code>
<code>swap volume, remove LVM logical volume link</code>	<code>lvrmboot(1M)</code>
<code>swapcontext()</code> - manipulate user contexts	<code>makecontext(2)</code>
<code>swapinfo</code> - system paging space information	<code>swapinfo(1M)</code>
<code>swapon</code> - enable device or file system for paging	<code>swapon(1M)</code>
<code>swapon()</code> - add swap device for interleaved paging and swapping	<code>swapon(2)</code>
<code>swapping, add swap device for interleaved</code>	<code>swapon(2)</code>
<code>swapping, enable device or file system</code>	<code>swapon(1M)</code>
<code>swapping, file system</code>	<code>swapon(2)</code>
<code>swask</code> - ask for user response for SD-UX	<code>swask(1M)</code>
<code>swconfig</code> - configure, unconfigure, reconfigure installed software	<code>swconfig(1M)</code>
<code>swcopy</code> - copy software products for subsequent installation or distribution	<code>swinstall(1M)</code>
<code>swgettools</code> - utility for retrieving the SD product from new SD media	<code>swgettools(1M)</code>
<code>swinstall</code> - install and configure software products	<code>swinstall(1M)</code>
<code>switch between screens</code>	<code>set_term(3X)</code>
<code>switch</code> - define switch statement	<code>csh(1)</code>
<code>switch to a new group</code>	<code>newgrp(1)</code>
<code>switched virtual circuit, clear X.25</code>	<code>clrsvc(1M)</code>
<code>swjob</code> - display job information and remove jobs	<code>swjob(1M)</code>
<code>swlist</code> - display information about software products	<code>swlist(1M)</code>
<code>swmodify</code> - modify software products in a target root or depot	<code>swmodify(1)</code>
<code>swpackage</code> - package software products into a target depot or tape	<code>swpackage(1M)</code>
<code>swpackage</code> - product specification file (PSF) format	<code>swpackage(4)</code>
<code>swreg</code> - register or unregister depots and roots	<code>swreg(1M)</code>
<code>swremove</code> - unconfigure and remove software products	<code>swremove(1M)</code>
<code>swverify</code> - verify software products	<code>swverify(1M)</code>
<code>symbol and line number information, strip from an object file</code>	<code>strip(1)</code>

Index

All Volumes

Description

Entry Name(Section)

symbol table for object code file, print	nm(1)
symbol table structure format	nlist(4)
symbol table, regenerate archive	ranlib(1)
symbol, get information for a global kernel symbol	getksym(2)
symbol, look up in shared library	shl_load(3X)
symbolic (soft) file link	symlink(4)
symbolic constants, standard structures and	unistd(5)
symbolic link	glossary(9)
symbolic link to a file, make a	symlink(2)
symbolic link, get status	lstat(2)
symbolic link, read value of	readlink(2)
symbolic links between files or directories, create	ln(1)
symbolic translation file for localedef scripts	charmap(4)
symbolically link NIS+ objects	nisln(1)
symlink - symbolic file link	symlink(4)
symlink() - make symbolic link to a file	symlink(2)
sync - flush unwritten system buffers to disk	sync(1M)
sync the nis+ password table with the nis+ trusted table	ttsyncd(1M)
sync() - update super-block	sync(2)
syncer - periodically sync for file system integrity	syncer(1M)
synchronise a window with its parents or children	syncok(3X)
synchronize a file's in-core state with its state on disk	fsync(2)
synchronize a mapped file	msync(2)
synchronize asynchronous I/O	aio_fsync(2)
synchronize stale logical volume mirrors in LVM volume groups	vgsync(1M)
synchronize stale mirrors in LVM logical volumes	lvsync(1M)
synchronize the system clock, correct the time to	adjtime(2)
synchronize transport library for transport endpoint (X/OPEN TLI-XTI)	t_sync(3)
synchronous I/O multiplexing	select(2)
syncok() - synchronise a window with its parents or children	syncok(3X)
syntax, a shell (command interpreter) with C-like	csh(1)
syntax, gated configuration file	gated.conf(4)
sysconf() - get configurable system variables	sysconf(2)
sysdef - display system definition	sysdef(1M)
sysfs() - get file system type info	sysfs(2)
syslog() - write message onto system log file	syslog(3C)
syslogd - log system messages	syslogd(1M)
system accounting, shell procedures for	acctsh(1M)
system activity daily report package	sa1(1M)
system activity reporter	sar(1M)
system administration manager	sam(1M)
system administrator	passwd(1)
system alias database	elm(1)
system alias text file	elm(1)
system alias: install new elm aliases	newaliases(1)
system aliases, elm , verify and display	elmalias(1)
system asynchronous I/O	glossary(9)
system buffers, flush unwritten buffers to disk	sync(1M)
system buffers, periodically flush unwritten buffers to disk	syncer(1M)
system call	glossary(9)
system call or event audit status, change or display	audevent(1M)
system call, data returned by stat/fstat/lstat	stat(5)
system calls and events currently being audited, get	getevent(2)
system calls and events to be audited	setevent(2)
system calls, BSD-4.2-compatible kill() , sigvec() , and signal()	bsdproc(3C)
system calls, BSD-4.2-compatible kill() , sigvec() , and signal()	killpg(2)
system calls, introduction	intro(2)
system calls, STREAMS	stream(2)
system clock date and time, get	gettimeofday(2)
system clock, correct the time to synchronize the	adjtime(2)
system clock, display current date and time or set to new value	date(1)

Description	Entry Name(Section)
system configuration file, change	ch_rc(1M)
system configuration/DHCP support script	auto_parms(1M)
system console	glossary(9)
system console interface special file	console(7)
system data types, primitive	types(5)
system database of automatically pushed STREAMS modules, manage	autopush(1M)
system default database entry, manipulate	getprdfent(3)
system default database file, trusted system	default(4)
system definition, display	sysdef(1M)
system diagnostic messages, collect to form error log	dmesg(1M)
system error messages	perror(3C)
system file, create a kernel	create_sysfile(1M)
system	glossary(9)
system information, display	uname(1)
system information, get	uname(2)
system loader, initial	isl(1M)
system log, control	syslog(3C)
system log, make entries in	logger(1)
system login data, display	logins(1M)
system maintenance commands and application programs, introduction	intro(1M)
system messages, log	syslogd(1M)
system name, display/set	uname(1)
system name, get/set	uname(2)
system nice value	renice(1)
system of process's expected paging behavior, advise	madvise(2)
system over LAN, log in on another	vt(1)
system paging space information	swapinfo(1M)
system parameter, query, set, or reset system parameter	kmtune(1M)
system physical environment daemon	envd(1M)
system process	glossary(9)
system resource consumption limit, get or set	getrlimit(2)
system scheduling priority	renice(1)
system shells, overview of various	sh(1)
system shutdown, turn accounting off for	acctsh(1M)
system startup and shutdown, timed	power_onoff(1M)
system startup, start accounting process at	acctsh(1M)
system status server	rwhod(1M)
system support for local network packet routing	routing(7)
system to UNIX system command execution, UNIX	uux(1)
system to UNIX system file copy, public UNIX	uuto(1)
system up time, show	uptime(1)
system users, list current	who(1)
system variables, get configurable	sysconf(2)
system() - issue a shell command	system(3S)
system, auditing	see audit
system, boot	reboot(2)
system, call another (UNIX); terminal emulator	cu(1)
system, display and update information about top processes on	top(1)
system, display information	uname(1)
system, get information	uname(2)
system, list users currently on the	users(1)
system, log in to	login(1)
system, scan for disk arrays	arrayscan(1M)
system, scan the I/O	ioscan(1M)
system, send LP request to remote	rlp(1M)
system, set node name	uname(1)
system, set node name	uname(2)
system, set or display name of current host	hostname(1)
system, trusted system device assignment database file	devassign(4)
system-calls error indicator	errno(2)
system-wide clock, get current value of	getclock(3C)

Index

All Volumes

Description

Entry Name(Section)

system-wide clock, set value of	setclock(3C)
system-wide sendmail aliases, print	praliases(1)
system: configure and build an HP-UX system	config(1M)
system: configure crash dumps	crashconf(1M)
system: configure system crash dumps	crashconf(2)
systems with label checking, copy file	volcopy(1M)
systems with label checking, copy file	volcopy_hfs(1M)
systems, uucp, list names of known	uucp(1)
sys_errlist() - system error messages	perror(3C)
sys_nerr() - system error messages	perror(3C)
table /etc/mnttab, establish mount	setmnt(1M)
table functions, NIS+	nis_tables(3N)
table preprocessor for nroff	tbl(1)
table, eliminate duplicate entries in a	lsearch(3C)
table, linear search for entry; optional update if missing	lsearch(3C)
table, mounted file system	mnttab(4)
table, symbol, for object code file, print	nm(1)
table, time zone adjustment, for date and ctime	tztab(4)
table, translate host-to-name-server file format	hosts_to_named(1M)
tables manually, manipulate routing	route(1M)
tables, binary search routine for sorted	bsearch(3C)
tables, generate iconv translation	genxlt(1)
tables, hash search, manage	hsearch(3C)
tables, populate NIS+ tables in a NIS+ domain	nispopulate(1M)
tabs - set tabs on a terminal	tabs(1)
tabs, convert to spaces and vice versa	expand(1)
tacct (total accounting) file, print any	acctsh(1M)
taddr2uaddr() - generic transport name-to-address translation	netdir(3N)
tags file, create a	ctags(1)
tail - get lines from last part of a file	tail(1)
tails the mail log	mtail(1M)
talk - talk to another user	talk(1)
talk to another user	talk(1)
talk to the keyser process	keyenvoy(1M)
talkd - remote user communication server	talkd(1M)
tan() - tangent function	tan(3M)
tand() - tangent function (degrees)	tand(3M)
tandf() - tangent function (float, degrees)	tand(3M)
tanf() - tangent function (float)	tan(3M)
tangent function (degrees)	tand(3M)
tangent function	tan(3M)
tangent function, inverse hyperbolic	atanh(3M)
tangent functions, hyperbolic	tanh(3M)
tanh() , tanhf() - hyperbolic tangent functions	tanh(3M)
tanhf() , tanh() - hyperbolic tangent functions	tanh(3M)
tape archive format, tar	tar(4)
Tape Cartridge I/O Utility, CS/80	tcio(1)
tape cartridge, partition DDS	mediainit(1)
tape device access drivers, cartridge	ct(7)
tape dump and restore protocol module, remote magnetic	rmt(1M)
tape file archiver	tar(1)
tape files: convert, reblock, translate, and copy	dd(1)
tape I/O, faster	ftio(1)
tape, package software products into	swpackage(1M)
tape, SCSI sequential access device driver	scsi_tape(7)
tape1, magnetic tape interface for stape, tape2 and	mt(7)
tape2, magnetic tape interface for stape,tape1, and	mt(7)
tar tape archive format	tar(4)
tar - tape file archiver	tar(1)
target depot, package software products into	swpackage(1M)
target process, force to run serially with other processes	serialize(1)

Description	Entry Name(Section)
target root, modify software products in depot or	swmodify(1M)
tbl - table preprocessor for nroff	tbl(1)
tbl, nroff / troff , and neqn constructs, remove	deroff(1)
tcdrain() : tty line control function	tccontrol(3C)
tcflow() : tty line control function	tccontrol(3C)
tcflush() : tty line control function	tccontrol(3C)
tcgetattr() - get tty device operating parameters	tcattribute(3C)
tcgetpgrp() : get foreground process group ID	tcgetpgrp(3C)
tcgetsid() - get terminal session ID	tcgetsid(3C)
tcio - Command Set 80 Cartridge Tape Utility	tcio(1)
TCP connection, identify user	idlookup(1)
TCP - Internet Transmission Control Protocol	TCP(7P)
TCP/IP IDENT protocol server	identd(1M)
tcsendbreak() : tty line control function	tccontrol(3C)
tcsetattr() - set tty device operating parameters	tcattribute(3C)
tcsetpgrp() - set foreground process group ID	tcsetpgrp(3C)
tdelete() - delete a node from a binary search tree	tsearch(3C)
tee - pipe fitting	tee(1)
tellldir() - get current location of named directory stream	directory(3C)
telm - STREAMS Telnet (pseudo-terminal) driver	tels(7)
Telnet drivers, STREAMS	tels(7)
Telnet port identification feature, dedicated ports file used by DDFA software and	dp(4)
TELNET protocol server	telnetd(1M)
TELNET protocol, user interface to the	telnet(1)
telnet - user interface to the TELNET protocol	telnet(1)
telnetd - TELNET protocol server	telnetd(1M)
tels - STREAMS Telnet (pseudo-terminal) driver	tels(7)
temperature environment, handle over-	envd(1M)
tempnam() - create a name for a temporary file	tmpnam(3S)
temporary (unique) file name, make a	mktemp(3C)
temporary file, create a name for	tmpnam(3S)
temporary file, create a	tmpfile(3S)
temporary file, make a name for a	mktemp(1)
term - format of compiled terminfo file	term(4)
term - terminal capabilities	term_c(4)
term.h - terminal capabilities	term_c(4)
termattrs() - get supported terminal video attributes	termattrs(3X)
termcap database emulation	tgetent(3X)
termcap description, convert into a terminfo description	captoinfo(1M)
termcap() access routines, emulate /etc/	termcap(3X)
terminal affiliation	glossary(9)
terminal baud rate, get	baudrate(3X)
terminal block mode interface	blmode(7)
terminal block-mode library interface	blmode(3C)
terminal capabilities	term_c(4)
terminal capabilities, disable use of	filter(3X)
terminal capabilities, get from terminfo database	tput(1)
terminal capability database	terminfo(4)
terminal characteristics for cue , set	cuegetty(1M)
terminal connection, set terminal type, modes, speed and line discipline for 2-way line	uugetty(1M)
terminal control database entry, manipulate	getprtcnt(3)
terminal control database	ttys(4)
Terminal Controller Device File Access software, Data Communications and	ddfa(7)
terminal device	glossary(9)
terminal device, control (Bell Version 6 compatibility)	stty(2)
terminal echo, enable/disable	echo(3X)
terminal emulator, IBM 3179G/3192G, 3270, 3777	sna(1)
terminal emulator, keyboard mapping	itemap(1M)
terminal emulator; call another (UNIX) system	cu(1)
terminal	glossary(9)
terminal I/O, block-mode library interface for	blmode(3C)

Description

Entry Name(Section)

terminal insert and delay capability, query functions	has_ic(3X)
terminal interface device file, process-controlling	tty(7)
terminal interface, extended general	termiox(7)
terminal interface, general	termio(7)
terminal interface, Version 6/PWB-compatible	sttyv6(7)
terminal line connection, establish an out-bound	dial(3C)
terminal line discipline module, STREAMS	ldterm(7)
terminal mode, save/restore	resetty(3X)
terminal name, get	termname(3X)
terminal or pseudo-terminal, get name of user's	tty(1)
terminal output control functions	clearok(3X)
terminal refresh, immediate, enable/disable	immedok(3X)
terminal screen, clear	clear(1)
terminal screen, number of columns	COLS(3X)
terminal screen, number of lines	LINES(3X)
terminal session ID, get	tcgetsid(3C)
Terminal Session Manager state information, get	tsm.info(1)
Terminal Session Manager	tsm(1)
terminal session, record typescript of	script(1)
terminal session, start	login(1)
terminal settings and datacomm line speed used by getty	gettydefs(4)
terminal type, modes, speed and line discipline, set for 2-way line	uugetty(1M)
terminal type, modes, speed, and line discipline, set	getty(1M)
terminal video attributes, get supported ones	termattrs(3X)
terminal, convert underscores to underlining on	ul(1)
terminal, deny or permit <i>write(1)</i> messages from other users to	mesg(1)
terminal, find name of	ttyname(3C)
terminal, generate file name of controlling	ctermid(3S)
terminal, get a multi-byte character length limited string from	getnstr(3X)
terminal, get a multi-byte character string	getstr(3X)
terminal, get a single-byte character	getch(3X)
terminal, get a wide character from	get_wch(3X)
terminal, get an array of wide characters and function key codes	getn_wstr(3X)
terminal, get name of user logged in on this terminal	getlogin(3C)
terminal, get name	termname(3X)
terminal, get verbose description of	longname(3X)
terminal, identify type	ttytype(1)
terminal, information on current terminal	cur_term(3X)
terminal, initialize based on terminal type	tset(1)
terminal, lock against use by others	lock(1)
terminal, output attributes	vidattr(3X)
terminal, output commands to	putp(3X)
terminal, output cursor movement commands to	mvcur(3X)
terminal, pseudo-terminal driver	pty(7)
terminal, read password from while suppressing echo	getpass(3C)
terminal, remote, spawn getty to (call terminal)	ct(1)
terminal, set options for port	stty(1)
terminal, set tabs on a	tabs(1)
terminal-type data base for each <i>tty</i> port	ttytype(4)
terminal: spawn getty to (call) remote terminal	ct(1)
terminals, CRT, peruse file on	more(1)
terminals, HP 2640- and HP 2621-series, handle special functions of	hp(1)
terminals, VT320, VT100, Wyse60	cue(1)
terminals, VT320, VT100, Wyse60	sam(1M)
terminals, VT320, VT100, Wyse60	swinstall(1M)
terminals, VT320, VT100, Wyse60	swremove(1M)
terminate a per-process timer	rmtimer(3C)
terminate a process	kill(1)
terminate all active processes	killall(1M)
terminate all system processing	shutdown(1M)
terminate, cause the calling thread to terminate	pthread_exit(3T)

Description	Entry Name(Section)
terminate, wait for background processes to	wait(1)
terminate, wait for child process to	wait(2)
terminated, determine how last I/O read	io_get_term_reason(3I)
terminating PAM sessions	pam_open_session(3)
termination character on special file, set up I/O read	io_eol_ctl(3I)
termination of a specified thread, wait for	pthread_join(3T)
termination, register a function to be called at program	atexit(2)
terminfo data base compiler	tic(1M)
terminfo data base, de-compile	untic(1M)
terminfo database, interfaces	del_curterm(3X)
terminfo database, retrieve capabilities from	tigetflag(3X)
terminfo de-compiler	untic(1M)
terminfo description, convert from a termcap description	captoinfo(1M)
terminfo descriptions, compare or print out	infocmp(1M)
terminfo - printer, terminal, and modem capability database	terminfo(4)
termio - general terminal interface	termio(7)
termiox - extended general terminal interface	termiox(7)
termname() - get terminal name	termname(3X)
terms, glossary of	glossary(9)
term_attrs() - get supported terminal video attributes	termattrs(3X)
test contents of memory area	memory(3C)
test - evaluate condition for true or false	test(1)
test - evaluate conditional expression	csh(1)
test - evaluate conditional expression	ksh(1)
test - evaluate conditional expression	sh-posix(1)
test mass storage media or device for recording integrity and proper operation	mediainit(1)
test packets, send	ping(1M)
test, initialize, and manipulate signal sets	sigsetops(3C)
text allocation space of object files, print section sizes and	size(1)
text editor, line-oriented	ex(1)
text editor, screen-oriented	vi(1)
text editor: line-oriented	ed(1)
text editors	see editor
text file for CRT or line-printer output, format	nroff(1)
text file format specification	fspec(4)
text file	glossary(9)
text file, change or reformat a	newform(1)
text formatter	fnt(1)
text formatters	see adjust
text pattern scanning and processing language	awk(1)
text processors: remove preprocessor lines	unifdef(1)
text processors: reverse the left-to-right text character sequence in each line of a file	rev(1)
text string, read from message file	gettext(3C)
text, lock in memory	plock(2)
text, mathematical, preprocess and format for nroff	neqn(1)
tfind() - get data pointer for binary search tree	tsearch(3C)
tftp - trivial file transfer program	tftp(1)
tftpd - trivial file transfer protocol server	tftpd(1M)
tgetent() - get compiled terminfo data base entry into buffer	termcap(3X)
tgetent() - termcap database emulation	tgetent(3X)
tgetflag() - get availability of compiled boolean terminal capability	termcap(3X)
tgetflag() - termcap database emulation	tgetent(3X)
tgetnum() - get numeric value of compiled terminal capability	termcap(3X)
tgetnum() - termcap database emulation	tgetent(3X)
tgetstr() - get string value of compiled terminal capability	termcap(3X)
tgetstr() - termcap database emulation	tgetent(3X)
tgoto() - get compiled terminal cursor addressing string	termcap(3X)
tgoto() - termcap database emulation	tgetent(3X)
thread attribute object, initialize or destroy	pthread_attr_init(3T)
thread cancellation cleanup handler, register or remove	pthread_cleanup_pop(3T)
thread condition variable attributes object, initialize or destroy	pthread_condattr_init(3T)

Index

All Volumes

Description	Entry Name(Section)
thread condition variable, initialize or destroy	pthread_cond_init(3T)
thread condition variable, wait or timed wait	pthread_cond_wait(3T)
thread ID for the calling thread, obtain	pthread_self(3T)
thread identifiers, compare	pthread_equal(3T)
thread of execution, create	pthread_create(3T)
thread process-shared attribute, get or set	pthread_condattr_getpshared(3T)
thread, call an initialization routine only once, threads	pthread_once(3T)
thread, cancel execution of	pthread_cancel(3T)
thread-safe, get, set, or end network host entry	gethostent(3N)
thread-safe, get, set, or end protocol entry	getprotoent(3N)
thread-safe, get, set, or end service entry	getservent(3N)
thread-specific data associated with a key, get or set	pthread_getspecific(3T)
thread-specific data key, create or destroy	pthread_key_create(3T)
threads waiting on a condition variable, unblock	pthread_cond_signal(3T)
threads, POSIX.1c introduction	pthread(3T)
three-byte integers and long integers, convert between	l3tol(3C)
three-way differential file comparison	diff3(1)
three-way file merge	merge(1)
tic - terminfo data base compiler	tic(1M)
tigetflag() - retrieve capabilities from the terminfo database	tigetflag(3X)
tigetnum() - retrieve capabilities from the terminfo database	tigetflag(3X)
tigetstr() - retrieve capabilities from the terminfo database	tigetflag(3X)
time, times - print summary of time used by processes	ksh(1)
time, times - print summary of time used by processes	sh-posix(1)
time a command; report process accounting data and system activity	timeux(1)
time and date conversion	strptime(3C)
time and date, convert to string	ctime(3C)
time and date, convert to string	strftime(3C)
time and date, convert to wide-character string	wcsftime(3C)
time and date, get more precisely (Version 7 compatibility only)	ftime(2)
time and date, set via NTP	ntptime(1M)
time between reboots, evaluate	last(1)
time delay, execute commands after	at(1)
time interval, suspend execution for a	sleep(1)
time limit for I/O operations, set	io_timeout_ctl(3I)
time - measure time used to execute a command	time(1)
time - print accumulated shell and children process times	sh-bourne(1)
time - print summary of time used by shell and children	csh(1)
time profile, execution	profil(2)
time to leave, notify you when it is	leave(1)
time used, report CPU	clock(3C)
time zone adjustment table for date and ctime	tztab(4)
time zone display	date(1)
time() - get time	time(2)
time, convert user format date and	getdate(3C)
time, display current or set to new value	date(1)
time, get system clock	gettimeofday(2)
time, get	time(2)
time, set	stime(2)
timed wait or wait on a thread condition variable	pthread_cond_wait(3T)
timed, automatic system power on, and power off	power_onoff(1M)
timed-job execution daemon	cron(1M)
timeout limit for I/O operations, set	io_timeout_ctl(3I)
timeout() - control blocking on input	notimeout(3X)
timer operations	timers(2)
timer, allocate a per-process	mktimer(3C)
timer, free a per-process	rmtimer(3C)
timer, get value of a per-process	gettimer(3C)
timer, relatively arm a per-process	reltimer(3C)
timer, set or get value of process interval	getitimer(2)
timer, set the interval timer	ualarm(2)

Description	Entry Name(Section)
timer_create() - create timer	timers(2)
timer_delete() - delete timer	timers(2)
timer_getoverrun() - return timer expiration count	timers(2)
timer_gettime() - store timer expiration and reload value	timers(2)
timer_settime() - set timer expiration	timers(2)
times - print accumulated user and system process times	sh-bourne(1)
times() - get process and child process times	times(2)
times, file access and modification, set or update	utime(2)
times, get process and child process	times(2)
times, set file access and modification times	utimes(2)
TIMESHARE scheduling policy	rtsched(2)
timex - time a command; report process accounting data and system activity	timex(1)
timezone() - difference between UCT and local timezone	ctime(3C)
timod - STREAMS module for converting ioctl() calls into Transport Interface messages	timod(7)
tirdwr - STREAMS module for reads and writes by Transport Interface users	tirdwr(7)
TLI function, accept a connect request issued by a transport user	t_accept(3)
TLI function, acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
TLI function, allocate a library structure	t_alloc(3)
TLI function, bind address to transport endpoint	t_bind(3)
TLI function, close transport endpoint	t_close(3)
TLI function, disable transport endpoint	t_unbind(3)
TLI function, error message function	t_error(3)
TLI function, establish connection with another transport user	t_connect(3)
TLI function, establish transport endpoint	t_open(3)
TLI function, free library structure	t_free(3)
TLI function, get current state	t_getstate(3)
TLI function, get protocol-specific service information	t_getinfo(3)
TLI function, initiate orderly release at transport endpoint	t_sndrel(3)
TLI function, listen for connect request	t_listen(3)
TLI function, look at current event on transport endpoint	t_look(3)
TLI function, manage options for transport endpoint	t_optmgmt(3)
TLI function, receive confirmation from connect request	t_rcvconnect(3)
TLI function, receive data over connection	t_rcv(3)
TLI function, receive data unit from remote transport provider user	t_rcvdata(3)
TLI function, receive error information from unit data error indication	t_rcvuderr(3)
TLI function, retrieve disconnect information	t_rcvdis(3)
TLI function, send data or expedited data over a connection	t_snd(3)
TLI function, send data unit to transport user	t_sndudata(3)
TLI function, send user-initiated disconnect request	t_snddis(3)
TLI function, synchronize transport library for transport endpoint	t_sync(3)
tmpfile() - create a temporary file	tmpfile(3S)
tmpfile64() - file system API to support large files	fgetpos64(2)
tmpnam() - create a name for a temporary file	tmpnam(3S)
toascii() - translate characters to 7-bit ASCII	conv(3C)
tolower(), _tolower - translate characters to lower-case	conv(3C)
too many arguments (error); construct argument list(s) and execute command	xargs(1)
tool, EISA configuration	eisa_config(1M)
tools to process 16-bit characters	nl_tools_16(3X)
top - display and update information about top processes on system	top(1)
topological sort	tsort(1)
total accounting (tacct) file, print any	acctsh(1M)
total accounting files, merge or add	acctmerg(1M)
total accounting records, convert per-session records to	acctcon(1M)
touch - update access, modification, and/or change times of a file	touch(1)
touchline() - window refresh control functions	is_linetouched(3X)
touchwin() - window refresh control function	touchwin(3X)
toupper(), _toupper, - translate characters to upper-case	conv(3C)
tolower() - translate wide characters to lowercase	wconv(3C)
towupper() - translate wide characters to uppercase	wconv(3C)
tparam() - retrieve capabilities from the terminfo database	tigetflag(3X)
tput - query the terminfo database	tput(1)

Description	Entry Name(Section)
<code>tputs()</code> - decode terminal string padding information	termcap(3X)
<code>tputs()</code> - output commands to the terminal	putp(3X)
<code>tr</code> - translate selected characters	tr(1)
trace and log command, configure subsystem database	nettlconf(1M)
trace messages to standard output, write STREAMS event trace messages	strace(1M)
trace, multithreaded process	ttrace(2)
trace, process	ptrace(2)
tracing and logging binary files, format	netfint(1M)
tracing and logging configuration file, Network	nettlgen.conf(4)
tracing, control network	nettl(1M)
traffic statistics, print mail	mailstats(1)
transactions, list spooled <i>uucp</i> transactions grouped by transaction	uuls(1M)
transcription system	answer(1)
transfer files between systems	ftp(1)
transfer files using XMODEM-protocol	umodem(1)
transfer NIS database from NIS server to local node	ypxfr(1M)
transfer program, trivial file	tftp(1)
transfer protocol server, file	ftpd(1M)
transfer protocol server, trivial file	tftpd(1M)
transfer speed, inform system of required minimum I/O	io_speed_ctl(3I)
transfer <i>uucp</i> -system files in or out	uucico(1M)
translate character code to another code set	iconv(1)
translate character code to another code set	iconv(3C)
translate characters to upper-case, lower-case, or 7-bit ASCII	conv(3C)
translate host table to name server file format	hosts_to_named(1M)
translate selected characters	tr(1)
translate wide characters to uppercase or lowercase	wconv(3C)
translate, convert, reblock and copy a (tape) file	dd(1)
translation file for localedef scripts, symbolic	charmap(4)
translation tables to a readable format, dump iconv	dmpxlt(1)
translation tables, generate iconv	genxlt(1)
translation, class-dependent data, of ELF files	elf_xlate(3E)
translation, generic transport name-to-address	netdir(3N)
transport endpoint, acknowledge receipt of release (X/OPEN TLI-XTI)	t_rcvrel(3)
transport endpoint, disable (X/OPEN TLI-XTI)	t_unbind(3)
transport endpoint, establish (X/OPEN TLI-XTI)	t_open(3)
transport endpoint, initiate orderly release (X/OPEN TLI-XTI)	t_sndrel(3)
transport endpoint, manage options (X/OPEN TLI-XTI)	t_optmgmt(3)
transport endpoint, synchronize transport library (X/OPEN TLI-XTI)	t_sync(3)
transport files, schedule <i>uucp</i>	uusched(1M)
transport function argument structures (X/OPEN TLI-XTI), allocate library structure	t_alloc(3)
Transport Interface messages: STREAMS module for converting ioctl()	timod(7)
Transport Interface: STREAMS module for reads and writes	tirdwr(7)
transport library, synchronize (X/OPEN TLI-XTI)	t_sync(3)
transport provider user (X/OPEN TLI-XTI)	t_rcvdata(3)
transport user, accept connect request (X/OPEN TLI-XTI)	t_accept(3)
transport user, establish connection (X/OPEN TLI-XTI)	t_connect(3)
transport user, send data unit (X/OPEN TLI-XTI)	t_sndudata(3)
trap enable bits: getting	fegettrapenable(3M)
trap enable bits: setting	fesettrapenable(3M)
<code>trap</code> - execute command upon receipt of signal	sh-bourne(1)
<code>trap</code> - trap specified signal	ksh(1)
<code>trap</code> - trap specified signal	sh-posix(1)
traverse (walk) a file tree, executing a function	ftw(3C)
traverse a binary search tree	tsearch(3C)
tree, manage a binary search	tsearch(3C)
tree, search directory tree for files	find(1)
tree, walk a file, executing a function	ftw(3C)
triangle, right, hypotenuse of a	hypot(3M)
trigonometric arccosine functions (degrees)	acosd(3M)
trigonometric arccosine functions	acos(3M)

Description	Entry Name(Section)
trigonometric arcsine functions (degrees)	asind(3M)
trigonometric arcsine functions	asin(3M)
trigonometric arctangent functions (degrees)	atand(3M)
trigonometric arctangent functions	atan(3M)
trigonometric arctangent-and-quadrant functions	atan2(3M)
trigonometric arctangent-and-quadrant functions (degrees)	atan2d(3M)
trigonometric cosine functions (degrees)	cosd(3M)
trigonometric cosine functions	cos(3M)
trigonometric sine functions (degrees)	sind(3M)
trigonometric sine functions	sin(3M)
trigonometric tangent functions (degrees)	tand(3M)
trigonometric tangent functions	tan(3M)
trivial file transfer program	tftp(1)
trivial file transfer protocol server	tftpd(1M)
troff/nroff files, check	checknr(1)
troff/nroff, tbl, and neqn constructs, remove	deroff(1)
true - do nothing and return zero exit status	true(1)
true, wait until the requested status condition becomes	hpib_status_wait(3I)
true/false evaluate condition for	test(1)
trunc() - truncation function	trunc(3M)
truncate an existing file to zero for rewriting	creat(2)
truncate text from beginning of line to specified column	newform(1)
truncate text line to specified maximum length	newform(1)
truncate() - truncate a file to a specified length	truncate(2)
truncate64() - file system API to support large files	creat64(2)
truncation function	trunc(3M)
trusted system device assignment database file	devassign(4)
trusted system password database, get entry from	getspent(3C)
trusted system, check if converted	iscomsec(2)
trusted system, system default database file	default(4)
trusted system, system device assignment database file	devassign(4)
tsearch() - build and access a binary search tree	tsearch(3C)
tset - terminal-dependent initialization	tset(1)
tsm , add or remove a printer for use with	tsm.lpadmin(1M)
tsm - Terminal Session Manager	tsm(1)
tsm.command - send commands to Terminal Session Manager	tsm.command(1)
tsm.info - get Terminal Session Manager state information	tsm.info(1)
tsm.lpadmin - add or remove a printer for use with tsm	tsm.lpadmin(1M)
tsort - topological sort	tsort(1)
tttrace request, wait for	tttrace_wait(2)
ttrace() - tracing facility for multithreaded process	ttrace(2)
ttrace_wait - wait for ttrace() request	ttrace_wait(2)
ttsyncd - daemon to maintain the nis+ password table in sync with the nis+ trusted table	ttsyncd(1M)
tty baud rate, set or get	cfspeed(3C)
tty device operating parameters, get or set	tcattribute(3C)
tty - get the name of the user's terminal or pseudo-terminal	tty(1)
tty	glossary(9)
tty line control functions	tccontrol(3C)
tty port, terminal-type data base for each	ttytype(4)
tty - process-controlling terminal interface device file	tty(7)
ttyname() , isatty() - find name of a terminal	ttyname(3C)
ttys and users, indicate last logins of	last(1)
ttys - terminal control database	ttys(4)
ttyslot() - find the slot in the utmp() file of the current user	ttyslot(3C)
ttytype - data base of terminal-type for each tty port	ttytype(4)
ttytype - terminal identification program	ttytype(1)
tun - IP network tunnel driver	tun(4)
tunable parameter, default_disk_ir	mount(1M)
tunable parameters, display system	sysdef(1M)
tunefs - tune up an existing HFS file system	tunefs(1M)
tuning, network	ndd(1M)

Index All Volumes

Description	Entry Name(Section)
tunnel driver, IP network	tun(4)
turnacct - turn process accounting on or off	acctsh(1M)
twalk() - traverse a binary search tree	tsearch(3C)
two files, compare	cmp(1)
two sorted files, reject/select lines common to	comm(1)
type attribute, get or set	pthread_mutexattr_getpshared(3T)
type control characters, how to	ascii(5)
type - show interpretation of <i>name</i> as if a command	sh-bourne(1)
type, classify characters according to	ctype(3C)
type, classify characters according to	wctype(3C)
type, determine file	file(1)
type, determine file system	fstyp(1M)
typeahead() - control checking for typeahead	typeahead(3X)
typeahead, control checking for	typeahead(3X)
types - primitive system data types	types(5)
types, system data primitives	types(5)
typescript of terminal session, record	script(1)
typeset - control leading blanks and parameter handling	ksh(1)
typeset - control leading blanks and parameter handling	sh-posix(1)
TZ environment variable	date(1)
tzname() - name of local timezone	ctime(3C)
tzset() - initialize <i>timezone()</i> , <i>daylight()</i> , and <i>tzname()</i> using TZ variable	ctime(3C)
tztab - time zone adjustment table for <i>date</i> and <i>ctime</i>	tztab(4)
t_accept() - accept a connect request issued by a transport user (X/OPEN TLI-XTI)	t_accept(3)
t_alloc() - allocate library structure for transport function argument structures (X/OPEN TLI-XTI)	t_alloc(3)
t_bind() - bind address to transport endpoint (X/OPEN TLI-XTI)	t_bind(3)
t_close() - close transport endpoint (X/OPEN TLI-XTI)	t_close(3)
t_connect() - establish connection with another transport user (X/OPEN TLI-XTI)	t_connect(3)
t_error() - error message function (X/OPEN TLI-XTI)	t_error(3)
t_free() - free memory for library structure (X/OPEN TLI-XTI)	t_free(3)
t_getinfo() - get protocol-specific service information (X/OPEN TLI-XTI)	t_getinfo(3)
t_getprotaddr() - get protocol address (X/OPEN XT)	t_getprotaddr(3)
t_getstate() - get current state (X/OPEN TLI-XTI)	t_getstate(3)
t_listen() - listen for connect request (X/OPEN TLI-XTI)	t_listen(3)
t_look() - look at current event on transport endpoint (X/OPEN TLI-XTI)	t_look(3)
t_open() - establish transport endpoint (X/OPEN TLI-XTI)	t_open(3)
t_optmgmt() - manage options for transport endpoint (X/OPEN TLI-XTI)	t_optmgmt(3)
t_rcv() - receive normal or expedited data sent over connection (X/OPEN TLI-XTI)	t_rcv(3)
t_rcvconnect() - receive confirmation from connect request (X/OPEN TLI-XTI)	t_rcvconnect(3)
t_rcvdis() - retrieve information from disconnect (X/OPEN TLI-XTI)	t_rcvdis(3)
t_rcvrel() - acknowledge receipt of release indication at transport endpoint (X/OPEN TLI-XTI)	t_rcvrel(3)
t_rcvudata() - receive data unit from remote transport provider user (X/OPEN TLI-XTI)	t_rcvudata(3)
t_rcvuderr() - receive error information from unit data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)
t_snd() - send data or expedited data over a connection (X/OPEN TLI-XTI)	t_snd(3)
t_snddis() - send user-initiated disconnect request (X/OPEN TLI-XTI)	t_sndis(3)
t_sndrel() - initiate orderly release at transport endpoint (X/OPEN TLI-XTI)	t_sndrel(3)
t_sndudata() - send data unit to transport user (X/OPEN TLI-XTI)	t_sndudata(3)
t_strerror() - produce error message string (X/OPEN - XT)	t_strerror(3)
t_sync() - synchronize transport library for transport endpoint (X/OPEN TLI-XTI)	t_sync(3)
t_unbind() - disable transport endpoint (X/OPEN TLI-XTI)	t_unbind(3)
u370 - is processor an IBM 370?	machid(1)
u3b - is processor a U3B?	machid(1)
u3b10 - is processor a U3B10?	machid(1)
u3b2 - is processor a U3B2?	machid(1)
u3b5 - is processor a U3B5?	machid(1)
uaddr2taddr() - generic transport name-to-address translation	netdir(3N)
ualarm() - set the interval timer	ualarm(2)
udp - Internet user datagram protocol	UDP(7P)
udpublickey - updates the <i>publickey</i> database file and NIS map	udpublickey(1M)
UID, get name from (obsolete)	getpw(3C)
u1 - do underlining on terminal	ul(1)

Description	Entry Name(Section)
ulckpwwdf() - control access to /etc/passwd file	lckpwwdf(3C)
ulimit - impose file size limit for child processes	sh-bourne(1)
ulimit - set size or time limits	ksh(1)
ulimit - set size or time limits	sh-posix(1)
ulimit() - get or set file size limits and break value	ulimit(2)
ultoa() - convert unsigned long integer to ASCII decimal	ltostr(3C)
ultoa_r() - convert unsigned long integer to ASCII decimal (MT-Safe)	ltostr(3C)
ultostr() - convert unsigned long integer to string	ltostr(3C)
ultostr_r() - convert unsigned long integer to string (MT-Safe)	ltostr(3C)
umask - set access permissions mode mask for file-creation	umask(1)
umask - set permissions mask for creating new files	csh(1)
umask - set permissions mask for creating new files	ksh(1)
umask - set permissions mask for creating new files	sh-bourne(1)
umask - set permissions mask for creating new files	sh-posix(1)
umask() - set and get file creation (permissions) mask	umask(2)
umodem - XMODEM-protocol file transfer program	umodem(1)
umount - unmount a file system (generic)	mount(1M)
umount - unmount CDFS file systems	mount_cdfs(1M)
umount - unmount HFS file systems	mount_hfs(1M)
umount - unmount NFS file systems	mount_nfs(1M)
umount - unmount VxFS file system	mount_vxfs(1M)
umount() - unmount a file system	umount(2)
umountall - unmount multiple file systems	mountall(1M)
unalias - discard specified alias	csh(1)
unalias - discard specified alias	ksh(1)
unalias - discard specified alias	sh-posix(1)
uname - display information about computer system; set node name (system name)	uname(1)
uname() - get information about computer system	uname(2)
unbiased exponent function	ilogb(3M)
unlock one or all threads waiting on a conditional variable	pthread_cond_signal(3T)
uncompact previously compacted Huffman coded files (see <i>pack(1)</i>)	compact(1)
uncompact - uncompact Huffman coded files (see <i>pack(1)</i>)	compact(1)
uncompile terminfo data base	untic(1M)
uncompress, compress, zcat - compress or expand data	compress(1)
uncompressdir, compressdir - compress or expand files in a directory	compress(1)
unconfigure, reconfigure, configure installed software	swconfig(1M)
unctrl() - generate printable representation of a character	unctrl(3X)
underflow mode: getting floating-point	fegetflushzero(3M)
underflow mode: setting floating-point	fesetflushzero(3M)
underlining on terminal, convert underscores to	ul(1)
underscores, convert to underlining on terminal	ul(1)
undial() - establish an out-bound terminal line connection	dial(3C)
undo a previous get of an SCCS file	unget(1)
unexpand, expand - expand tabs to spaces, and vice versa	expand(1)
unexport and export directories to PFS clients	pfs_exportfs(1M)
unexport directories to NFS clients	exportfs(1M)
unget - undo a previous get of an SCCS file	unget(1)
ungetc() - push character back into input stream	ungetc(3S)
ungetch() - push a character onto the input queue	ungetch(3X)
ungetwc() - push wide character back into input stream	ungetwc(3C)
ungetwc_unlocked() - unlocked version of ungetwc()	ungetwc(3C)
unget_wch() - push a character onto the input queue	ungetch(3X)
unhash - disable use of internal hash tables	csh(1)
unifdef - remove preprocessor lines	unifdef(1)
uninterpreted file contents, retrieve for ELF files	elf_rawfile(3E)
Uninterruptible Power System (UPS), monitor daemon	ups_mond(1M)
Uninterruptible Power System monitor configuration file	ups_conf(4)
uniq - report adjacent repeated lines in a file	uniq(1)
unique (usually temporary) file name, make a	mktemp(3C)
unistd.h - standard structures and symbolic constants	unistd(5)
unit data error indication (X/OPEN TLI-XTI)	t_rcvuderr(3)

Index

All Volumes

Description	Entry Name(Section)
units - convert units of measure	units(1)
units of measure, convert	units(1)
UNIX - local communication domain protocol	UNIX(7P)
UNIX system to UNIX system command execution	uux(1)
UNIX system to UNIX system file copy, public	uuto(1)
unlink a message queue	mq_unlink(2)
unlink a named semaphore	sem_unlink(2)
unlink a shared memory object	shm_unlink(2)
unlink - execute unlink() system call without error checks	link(1M)
unlink - remove directory entry; delete file	unlink(2)
unlink() system call, execute without error checks	link(1M)
unload a kernel module on demand	moduload(2)
unload shared library	shl_load(3X)
unlock a mutex	pthread_mutex_unlock(3T)
unlock a POSIX semaphore	sem_post(2)
unlock a read-write lock	pthread_rwlock_unlock(3T)
unlock a semaphore	msem_unlock(2)
unlock a STREAMS pty master/slave pair	unlockpt(3C)
unlock access to /etc/passwd file	lckpwwdf(3C)
unlock memory segment	munlock(2)
unlock or lock an I/O interface	io_lock(3I)
unlock process virtual address space	munlockall(2)
unlockpt() - unlock a STREAMS pty master and slave pair	unlockpt(3C)
unmap a mapped region	munmap(2)
unmount a file system (generic)	mount(1M)
unmount a file system	umount(2)
unmount and mount CD-ROM file systems	pfs_mount(1M)
unmount CDFS file systems	mount_cdfs(1M)
unmount HFS file systems	mount_hfs(1M)
unmount multiple file systems	mountall(1M)
unmount NFS file systems	mount_nfs(1M)
unmount VxFS file system	mount_vxfs(1M)
unpack - expand Huffman coded files created by <i>pack</i> (see <i>compact</i> (1))	pack(1)
unprintable characters in a file, make visible or invisible	vis(1)
unregister or register depots and roots	swreg(1M)
unset - remove definition/setting of flags and arguments	csh(1)
unset - remove definition/setting of options and arguments	ksh(1)
unset - remove definition/setting of options and arguments	sh-bourne(1)
unset - remove definition/setting of options and arguments	sh-posix(1)
unsetenv - remove variable from environment	csh(1)
unsigned long integer to string, convert	ltostr(3C)
untic - terminfo de-compiler	untic(1M)
until - execute commands until expression is non-zero	ksh(1)
until - execute commands until expression is nonzero	sh-posix(1)
untouchwin() - window refresh control functions	is_linetouched(3X)
unused DOS disk clusters, report number of	dosdf(1)
unwritten system buffers, flush to disk	sync(1M)
unwritten system buffers, periodically flush to disk	syncer(1M)
update access, modification, and/or change times of a file	touch(1)
update an ELF descriptor	elf_update(3E)
update backup LVM volume group configuration file	vgcfgbackup(1M)
update default kernel files or specified kernel modules	kmupdate(1M)
update execution time limit	rtsched(2)
update file access and modification times	utime(2)
update files on remote hosts	rdist(1)
update secure password facility	pwconv(1M)
update status, line, functions	redrawwin(3X)
update super-block	sync(2)
update table if entry missing after search	lsearch(3C)
update the public keys in a NIS+ directory object	nisupdkeys(1M)
update user password in Network Information Service	yppasswd(3N)

Description	Entry Name(Section)
update, add, delete a kernel module	kminstall(1M)
update, install, or remove boot programs from a disk device	mkboot(1M)
update, maintain, and regenerate groups of programs	make(1)
updaters- configuration file for NIS updating	updaters(1M)
updates the publickey database file and NIS map	udpublickey(1M)
updating, configuration file for NIS	updaters(1M)
upgrade disk layout of VxFS file system	vxupgrade(1M)
upper-case, translate characters to	conv(3C)
uppercase, translate wide characters to	wconv(3C)
UPS monitor configuration file	ups_conf(4)
UPS, monitor daemon	ups_mond(1M)
upshifting	glossary(9)
ups_conf - HP PowerTrust UPS monitor configuration file	ups_conf(4)
ups_conf - Uninterruptible Power System (UPS) monitor configuration file	ups_conf(4)
ups_mond - HP PowerTrust monitor daemon	ups_mond(1M)
ups_mond - HP PowerTrust Uninterruptible Power System monitor daemon	ups_mond(1M)
ups_mond - Uninterruptible Power System monitor daemon	ups_mond(1M)
uptime - show how long system has been up	uptime(1)
usage quotas, disk	quota(5)
usage, summarize disk	du(1)
use, disable, of certain terminal capabilities	filter(3X)
user accounting file entry format for btmpt() , utmp() , and wtmp()	utmp(4)
user accounting information file	utmpx(4)
user accounting, daily accounting shell procedure	runacct(1M)
user alias database	elm(1)
user alias text file	elm(1)
user alias: install new elm aliases	newaliases(1)
user aliases, elm , verify and display	elmaliases(1)
user configuration files for pluggable authentication modules	pam_user.conf(4)
user context, get and set current	getcontext(2)
user contexts, manipulate	makecontext(2)
user credentials for an authentication service, modify and delete	pam_setcred(3)
user crontab file operations	crontab(1)
user datagram protocol, Internet	UDP(7P)
user disk usage quotas	quota(5)
User Environment (CUE), HP Character-Terminal	cue(1)
user environment variables	environ(5)
user format date and time, convert	getdate(3C)
user group access and identification file, grp.h	group(4)
user ID	glossary(9)
user id, effective current, print or display	whoami(1)
user ID, get real or effective	getuid(2)
user ID, set	setuid(2)
user IDs and names, print	id(1)
user IDs, set real and effective user IDs	setreuid(2)
user information in password file; used by finger , change	chfn(1)
user information lookup program	finger(1)
user information server, remote	fingerd(1M)
user interface for gated	gdc(1M)
user interface to the TELNET protocol	telnet(1)
user login data, display	logins(1M)
user login name, get character-string representation of	cuserid(3S)
user login name, obtain	logname(3C)
user login, display data	listusers(1M)
user name directory service, Internet	whois(1)
user name, PAM routine to retrieve	pam_get_user(3)
user name: in elm aliases	newaliases(1)
user of a particular TCP connection, identify	idlookup(1)
user or group IDs, set real, effective, and/or saved	setresuid(2)
user password in Network Information Service, update	yppasswd(3N)
user policy definition service module, PAM	pam_updb(5)

Index

All Volumes

Description	Entry Name(Section)
user processes currently using a file or file structure	fuser(1M)
user quotas, edit	edquota(1M)
user shells, get legal	getusershell(3C)
user's effective access rights to a file, get a	getaccess(2)
user's login environment, shell script to set up	profile(4)
user's terminal or pseudo-terminal, get name of	tty(1)
user, ask for user response for SD-UX	swask(1M)
user, bill fee to, based on system usage	acctsh(1M)
user, change login name to another	su(1)
user, change user's secure RPC key	chkey(1)
user, communicate interactively with another	write(1)
user, current, find the slot in the <code>utmp()</code> file of the	ttyslot(3C)
user, get name of user logged in on this terminal	getlogin(3C)
user, show last login date for each	acctsh(1M)
<code>user2netname()</code> - library routines for secure remote procedure calls	secure_rpc(3N)
user: print list of current system users	who(1)
<code>useradd</code> - add a user login on the system	useradd(1M)
<code>userdel</code> - delete a user login from the system	userdel(1M)
<code>usermod</code> - modify a user login on the system	usermod(1M)
username server, network	rusersd(1M)
users and processes, list current	whodo(1M)
users and ttys, indicate last logins of	last(1)
<code>users</code> - compact list of users currently on the system	users(1)
users currently on the system, list	users(1)
users on remote machines, return information about	rnusers(3N)
users over a network, write to all	rwall(1M)
users, notify of new mail in mailboxes	newmail(1)
users, remote, authorizing access on local host	hosts.equiv(4)
users, select for auditing	audusr(1M)
users: list current users and what they are doing	whodo(1M)
<code>use_env()</code> - specify source of screen size information	use_env(3X)
<code>usleep()</code> - suspend execution for an interval	usleep(2)
<code>ustat()</code> - get mounted file system statistics	ustat(2)
UTC (Coordinated Universal Time)	glossary(9)
UTC, Coordinated Universal Time	date(1)
utility for searching NIS+ tables	nismatch(1)
utility	glossary(9)
utility options, parse	getopts(1)
utility program for SCCS commands	sccs(1)
<code>utime()</code> - set or update file access and modification times	utime(2)
<code>utimes()</code> - set file access and modification times	utimes(2)
<code>utmp</code> file	login(1)
<code>utmp</code> record, write and include reason for writing	acct(1M)
<code>utmp()</code> , <code>wtmp()</code> , <code>btmp()</code> - <code>utmp</code> , <code>wtmp</code> , <code>btmp</code> user accounting file entry format	utmp(4)
<code>utmp()</code> file of the current user, find the slot in the	ttyslot(3C)
<code>utmp()</code> or <code>wtmp()</code> file, access	getut(3C)
<code>utmpname()</code> - change name of <code>utmp()</code> file being examined	getut(3C)
<code>utmpname_r()</code> - change name of <code>utmp()</code> file being examined	getut(3C)
<code>utmpx</code> - user accounting information file	utmpx(4)
<code>utmpx()</code> or <code>wtmp()</code> file, access	getutx(3C)
<code>uuccheck</code> - check the <code>uucp</code> directories and permissions file	uuccheck(1M)
<code>uucico</code> - transfer <code>uucp</code> -system files	uucico(1M)
<code>uuclean</code> - <code>uucp</code> spool directory clean-up	uuclean(1M)
<code>uucleanup</code> - <code>uucp</code> spool directory clean-up	uucleanup(1M)
<code>uucp</code> , <code>uulog</code> , <code>uuname</code> - UNIX system to UNIX system copy	uucp(1)
<code>uucp</code> -system files, transfer in or out	uucico(1M)
<code>uucp</code> and <code>uux</code> transactions summary log, access	uucp(1)
<code>uucp</code> or <code>uux</code> command requests from remote, execute on local system	uuxqt(1M)
<code>uucp</code> spool directory clean-up	uuclean(1M)
<code>uucp</code> spool directory clean-up	uucleanup(1M)
<code>uucp</code> status inquiry and job control	uustat(1)

Description	Entry Name(Section)
uucp subnetwork activity, monitor	uusub(1M)
uucp systems, list names of known	uucp(1)
uucp: check the uucp directories and permissions file	uucheck(1M)
uucp: list spooled <i>uucp</i> transactions grouped by transaction	uuls(1M)
uucp: schedule uucp transport files	uusched(1M)
uucp: set terminal type, modes, speed and line discipline for 2-way line	uugetty(1M)
uucp: show snapshot of the UUCP system	uusnap(1M)
uucp: <i>uucleanup</i> - uucp spool directory clean-up	uuclean(1M)
uucpd() - server for supporting UUCP over TCP/IP networks	uucpd(1M)
uudecode - decode a file encoded by <i>uuencode</i>	uuencode(1)
uuencode - encode a binary file for transmission by mailer	uuencode(1)
uuencode - format of a <i>uuencode(1)</i> -encoded file	uuencode(4)
uugetty - set terminal type, modes, speed and line discipline for 2-way line	uugetty(1M)
uulog - access uucp and uux transactions summary log	uucp(1)
uuls - list spooled <i>uucp</i> transactions grouped by transaction	uuls(1M)
uuname - list names of known uucp systems	uucp(1)
uupath, <i>mkuupath</i> - access and manage the pathalias database	uupath(1)
uupick - accept or reject files sent by <i>uuto</i>	uuto(1)
uusched - schedule uucp transport files	uusched(1M)
uusnap output, sort and embellish	uusnaps(1M)
uusnap - show snapshot of the UUCP system	uusnap(1M)
uusnaps - sort and embellish <i>uusnap</i> output	uusnaps(1M)
uustat - uucp status inquiry and job control	uustat(1)
uusub - monitor uucp subnetwork activity	uusub(1M)
uuto - public UNIX system to UNIX system file copy	uuto(1)
uux and uucp transactions summary log, access	uucp(1)
uux or uucp command requests from remote, execute on local system	uuxqt(1M)
uux - UNIX system to UNIX system command execution	uux(1)
uuxqt - execute remote uucp or uux command requests on local system	uuxqt(1M)
ux2dos, <i>dos2ux</i> - convert ASCII file format	dos2ux(1)
vacation - return "I am not here" indication	vacation(1)
val - validate an SCCS file	val(1)
validate an SCCS file	val(1)
validate whether physical page number was dumped	cr_isaddr(3)
validation procedures, perform PAM account	pam_acct_mgmt(3)
valloc() - allocate space on boundary aligned to sysconf value	malloc(3C)
value occurs, wait until a particular parallel poll	hplib_wait_on_ppoll(3I)
value of a per-process timer, get	gettimer(3C)
value of a symbolic link, read	readlink(2)
value of process interval timer, set or get	getitimer(2)
value of system-wide clock, get current	getclock(3C)
value of system-wide clock, set	setclock(3C)
value, change or add to environment	putenv(3C)
value, get or set file size limits and break	ulimit(2)
value, return integer absolute	abs(3C)
values and constants for programming, machine-dependent	values(5)
values in a Network Information Service map, print all	ypcat(1)
values - machine-dependent values	values(5)
values of selected keys in Network Information Service map, print the	ypmatch(1)
values, convert between host and network byte order	byteorder(3N)
values, get POSIX configuration	getconf(1)
values, get string-valued configuration	confstr(3C)
varargs argument list, print formatted output of a	vprintf(3S)
varargs argument, formatted input conversion to a	vscanf(3S)
<i>varargs.h</i> - macros for handling variable argument list	varargs(5)
variable argument list macros	stdarg(5)
variable argument list macros	varargs(5)
variable, environment, search environment list for value of	getenv(3C)
variables in the stable storage, display and modify	setboot(1M)
variables, configurable path name, get	pathconf(2)
variables, environment, print value of	printenv(1)

Index

All Volumes

Description	Entry Name(Section)
variables, system, get configurable	sysconf(2)
variables, user environment	environ(5)
vax - is processor a VAX?	machid(1)
vc - version control	vc(1)
vector, get option letter from argument	getopt(3C)
vedit - beginner's screen-oriented text editor	vi(1)
verbose description of current terminal, get	longname(3X)
verification tool, STREAMS	strvf(1M)
verifier, generic file system quota consistency	quotacheck(1M)
verifier, hfs file system quota consistency	quotacheck_nfs(1M)
verifier, VxFS file system quota consistency	quotacheck_vxfs(1M)
verify a remote user as a local user	rcmd(3N)
verify elm user and system aliases	elmalias(1)
verify integrity of crash dump	cr_verify(3)
verify integrity of mass storage media	mediainit(1)
verify internal revision numbers of HP-UX files	revck(1M)
verify LAN connectivity with link-level loopback	linkloop(1M)
verify Network License Servers are working	i4tv(1M)
verify program assertion	assert(3X)
Version 6/PWB-compatible terminal interface	sttyv6(7)
version control	vc(1)
version level of operating system, display	uname(1)
version level of operating system, get	uname(2)
version of an SCCS file, get a	get(1)
versions of an SCCS file, compare two	sccsdiff(1)
versions, coordinate ELF library and application	elf_version(3E)
vfork() - spawn new process (use fork() instead)	vfork(2)
vfprintf() - print formatted output of a varargs argument list	vprintf(3S)
vfscanf() - formatted input conversion to a varargs argument	vscanf(3S)
vfsmount() - mount a file system	vfsmount(2)
vgcfsbackup - create LVM volume group configuration backup file	vgcfsbackup(1M)
vgcfrestore - restore volume group configuration	vgcfrestore(1M)
vgchange - set LVM volume group availability	vgchange(1M)
vgcreate - create LVM volume group	vgcreate(1M)
vgdisplay - display information about LVM volume groups	vgdisplay(1M)
vgexport - export an LVM volume group and its associated logical volumes	vgexport(1M)
vgextend - extend an LVM volume group by adding physical volumes	vgextend(1M)
vgimport - import an LVM volume group onto the system	vgimport(1M)
vgreduce - remove physical volumes from an LVM volume group	vgreduce(1M)
vgremove - remove LVM volume group definition from the system	vgremove(1M)
vgscan - scan physical volumes for LVM volume groups	vgscan(1M)
vgsync - synchronize stale logical volume mirrors in LVM volume groups	vgsync(1M)
VHE home machine is not available, login when	vhe_altlog(1M)
vhe_altlog : login when VHE home machine is not available	vhe_altlog(1M)
vhe_list - Virtual Home Environment information file	vhe_list(4)
vhe_mounter : start the Virtual Home Environment (VHE)	vhe_mounter(1M)
vhe_u_mnt : perform NFS mount to remote file system	vhe_u_mnt(1M)
vi edit on the password file	vipw(1M)
vi - extended screen-oriented text editor	vi(1)
vidattr() - output attributes to terminal	vidattr(3X)
video attributes, terminal, get supported	termattrs(3X)
vidputs() - output attributes to terminal	vidattr(3X)
vid_attr() - output attributes to terminal	vidattr(3X)
vid_puts() - output attributes to terminal	vidattr(3X)
view or modify Access Control Lists	swacl(1M)
view - read-only screen-oriented text editor	vi(1)
viewing, saving SAM logfile tool	samlog_viewer
vipw - edit the password file	vipw(1M)
virtual circuit, X.25 switched, clear	clrsvc(1M)
Virtual Home Environment (VHE), start the	vhe_mounter(1M)
Virtual Home Environment information file	vhe_list(4)

Description	Entry Name(Section)
virtual memory statistics, report	vmstat(1)
virtual memory, map object into	mmap(2)
virtual terminal requests from other systems, respond to	vt daemon(1M)
vis - make unprintable and non-ASCII characters in a file visible	vis(1)
vline() - draw lines from single-byte characters and renditions	hline(3X)
vline_set() - draw lines from complex characters and renditions	hline_set(3X)
vmstat - report virtual memory statistics	vmstat(1)
volcopy - copy file systems with label checking	volcopy(1M)
volcopy - copy file systems with label checking	volcopy_hfs(1M)
volcopy - copy VxFS file system with label checking	volcopy_vxfs(1M)
volcopy_vxfs - copy VxFS file system with label checking	volcopy_vxfs(1M)
volume format, file system	fs(4)
volume group (LVM) configuration backup file, create or update	vgcfgbackup(1M)
volume group (LVM), change characteristics and access path of physical volume in	pvchange(1M)
volume group (LVM), check or repair a physical volume in	pvck(1M)
volume group (LVM), create logical volume in	lvcreate(1M)
volume group (LVM), create physical volume for use in	pvcreate(1M)
volume group (LVM), create	vgcreate(1M)
volume group (LVM), display information about physical volumes in	pvdisplay(1M)
volume group (LVM), extend by adding physical volumes	vgextend(1M)
volume group (LVM), import onto the system	vgimport(1M)
volume group (LVM), reduce by removing physical volumes	vgreduce(1M)
volume group (LVM), remove logical volumes from	lvremove(1M)
volume group and its associated logical volumes (LVM), export	vgexport(1M)
volume group availability (LVM), set	vgchange(1M)
volume group configuration, restore	vgcfgrestore(1M)
volume group definition (LVM), remove from the system	vgremove(1M)
volume group information file, LVM physical	lvmphys(4)
volume group	lvm(7)
volume groups (LVM), display information about	vgdisplay(1M)
volume groups (LVM), scan physical volumes for	vgscan(1M)
volume groups (LVM), synchronize stale logical volume mirrors	vgsync(1M)
volume header on LIF file, write	lifinit(1)
volume in LVM volume group, create logical	lvcreate(1M)
volume mirrors in LVM volume groups, synchronize stale logical	vgsync(1M)
volume number	glossary(9)
volumes from LVM volume group, remove logical	lvremove(1M)
vpfmt() - display message in standard format	pfmt(3C)
vprintf() - print formatted output of a varargs argument list	vprintf(3S)
vscanf() - formatted input conversion to a varargs argument	vscanf(3S)
vsprintf() - print formatted output of a varargs argument list	vprintf(3S)
vsprintf() - print formatted output of a varargs argument list	vprintf(3S)
vsscanf() - formatted input conversion to a varargs argument	vscanf(3S)
vt - log in on another system over LAN	vt(1)
vt requests from other systems, respond to	vt daemon(1M)
VT100 terminal	cue(1)
VT100 terminal	sam(1M)
VT100 terminal	swinstall(1M)
VT100 terminal	swremove(1M)
VT320 terminal	cue(1)
VT320 terminal	sam(1M)
VT320 terminal	swinstall(1M)
VT320 terminal	swremove(1M)
vt daemon - respond to vt requests	vt daemon(1M)
vwprintw() - print formatted output in a window	vwprintw(3X)
vwscanw() - convert formatted input from a window	vwprintw(3X)
vw_printw() - print formatted output in a window	vw_printw(3X)
vw_scanw() - convert formatted input from a window	vwprintw(3X)
vxdiskusg - generate disk accounting data of VxFS file system by user ID	vxdiskusg(1M)
vxdump - local incremental file system dump	vxdump(1M)
VxFS Advanced, get extent attributes	getext(1M)

Index All Volumes

Description	Entry Name(Section)
VxFS Advanced, set extent attributes	setext(1M)
VxFS file system control functions	vxfsio(7)
VxFS file system debugger	fsdb_vxfs(1M)
VxFS file system quota consistency checker	quotacheck_vxfs(1M)
VxFS file system volume, format	fs_vxfs(4)
VxFS file system with label checking	volcopy_vxfs(1M)
VxFS file system, cat	fscat_vxfs(1M)
VxFS file system, check and repair	fsck_vxfs(1M)
VxFS file system, construct	mkfs_vxfs(1M)
VxFS file system, disk accounting data by user ID	vxdiskusg(1M)
VxFS file system, generate pathnames from inode numbers	ncheck_vxfs(1M)
VxFS file system, list file names and statistics	ff_vxfs(1M)
VxFS file system, mount and unmount	mount_vxfs(1M)
VxFS file system, repair	fsck_vxfs(1M)
VxFS file system, report free disk blocks	df_vxfs(1M)
VxFS file system, resize or reorganize file system	fsadm_vxfs(1M)
VxFS file system: construct new file system	newfs_vxfs(1M)
VxFS inode, format	inode_vxfs(4)
vxfsio - VxFS file system control functions	vxfsio(7)
vxrestore - restore file system incrementally, local or across network	vxrestore(1M)
vxupgrade - upgrade disk layout of a VxFS file system	vxupgrade(1M)
w - show how long system has been up	uptime(1)
waddch() - add a single-byte character and rendition to a window and advance the cursor	addch(3X)
waddchnstr() - add length limited string of single-byte characters and renditions to a window	addchnstr(3X)
waddchstr() - add string of single-byte characters and renditions to a window	addchstr(3X)
waddnstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
waddnwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
waddstr() - add a string of multi-byte characters without rendition to a window and advance cursor	addnstr(3X)
waddwstr() - add a wide-character string to a window and advance the cursor	addnwstr(3X)
wadd_wch() - add a complex character and rendition to a window	add_wch(3X)
wadd_wchnstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
wadd_wchstr() - add an array of complex characters and renditions to a window	add_wchnstr(3X)
wait for a signal	sigsuspend(2)
wait for asynchronous I/O completion	aio_suspend(2)
wait for child process to change state	wait3(2)
wait for child process to change state	waitid(2)
wait for interrupt, atomically release blocked signals and	sigpause(2)
wait for the termination of a specified thread	pthread_join(3T)
wait for trace request	ttrace_wait(2)
wait or timed wait on a thread condition variable	pthread_cond_wait(3T)
wait until a particular parallel poll value occurs	hpib_wait_on_ppoll(3I)
wait until the requested status condition becomes true	hpib_status_wait(3I)
wait - wait for background processes	csh(1)
wait - wait for background processes to complete	wait(1)
wait - wait for child process	ksh(1)
wait - wait for child process	sh-posix(1)
wait - wait for process and report termination status	sh-bourne(1)
wait() - wait for child or traced process to stop or terminate	wait(2)
wait3() - wait for child process to change state	wait3(2)
waitid() - wait for child process to change state	waitid(2)
waitpid() - wait for child or traced process to stop or terminate	wait(2)
walk a file tree, executing a function	ftw(3C)
wall - write message to all users	wall(1M)
wattroff() - restricted window attribute control functions	attroff(3X)
wattron() - restricted window attribute control functions	attroff(3X)
wattrset() - restricted window attribute control functions	attroff(3X)
wattr_get() - window attribute control functions	attr_get(3X)
wattr_off() - window attribute control functions	attr_get(3X)
wattr_on() - window attribute control functions	attr_get(3X)

Description	Entry Name(Section)
wattr_set() - window attribute control functions	attr_get(3X)
wc - count words, lines, and bytes or characters in a file	wc(1)
WCHAR(), - put 8- or 16-bit character in memory	nl_tools_16(3X)
WCHARADV(), - put character in memory and advance pointer	nl_tools_16(3X)
wchgat() - change renditions of characters in a window	chgat(3X)
wclear() - clear a window	clear(3X)
wclrtoebot() - clear from cursor to end of window	clrtoebot(3X)
wclrtoeol() - clear from cursor to end of line	clrtoeol(3X)
wcolor_set() - window attribute control functions	attr_get(3X)
wconv() - translate wide characters	wconv(3C)
wscat(), wcsncat() - append wide string 2 to wide string 1	wcstring(3C)
wcschr(), wcsrchr() - get pointer to wide character in wide string	wcstring(3C)
wscmp(), wcsncmp() - compare two wide strings	wcstring(3C)
wscoll() - process wide string of text tokens	wcstring(3C)
wscpy(), wcsncpy() - copy wide string 2 to wide string 1	wcstring(3C)
wcscspn(), wcsspncpy() - find length of matching wide substrings	wcstring(3C)
wcsftime() - convert date and time to wide-character string	wcsftime(3C)
wcslen() - determine length of a wide string	wcstring(3C)
wcsprbrk() - find occurrence of wide character from wide string 2 in wide string 1	wcstring(3C)
wctod() - convert wide character string to double-precision number	wctod(3C)
wctok() - process wide string of text tokens	wcstring(3C)
wctok_r() - process wide string of text tokens	wcstring(3C)
wctol() - convert wide character string to long integer	wctol(3C)
wctombs() - convert sequence of codes corresponding to multibyte characters	multibyte(3C)
wctoul() - convert wide character string to long integer	wctol(3C)
wcswcs() - process wide string of text tokens	wcstring(3C)
wcsxfrm() - process wide string of text tokens	wcstring(3C)
wctomb() - number of bytes needed to represent multibyte character	multibyte(3C)
wctype() - classify characters according to type	wctype(3C)
wcursyncup() - synchronise a window with its parents or children	syncok(3X)
wdelch() - delete character from a window	delch(3X)
wdeleteln(), deleteln() - delete lines in window	deleteln(3X)
wechochar() - echo single-byte character and rendition to a window and refresh	echochar(3X)
wecho_wchar() - write a complex character and immediately refresh the window	echo_wchar(3X)
werase() - clear a window	clear(3X)
wgetch() - get a single-byte character from the terminal	getch(3X)
wgetnstr() - get a multi-byte character length limited string from the terminal	getnstr(3X)
wgetn_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
wgetstr() - get a multi-byte character string from the terminal	getstr(3X)
wget_wch() - get a wide character from a terminal	get_wch(3X)
wget_wstr() - get an array of wide characters and function key codes from a terminal	getn_wstr(3X)
what - get SCCS identification information from files	what(1)
whence - define interpretation of name as a command	ksh(1)
whence - define interpretation of name as a command	sh-posix(1)
whereis - locate source, binary, and/or manual files for program	whereis(1)
which - locate a program file including aliases and paths	which(1)
while - execute commands while expression is non-zero	cs(1)
while - execute commands while expression is non-zero	ksh(1)
while - execute commands while expression is nonzero	sh-posix(1)
whitespace	glossary(9)
whline() - draw lines from single-byte characters and renditions	hline(3X)
whline_set() - draw lines from complex characters and renditions	hline_set(3X)
who is logged in on local machines, show	rwho(1)
who is logged in on local network machines, determine	rusers(1)
who is my mail from?	from(1)
who - who is using the system	who(1)
whoami - print effective current user id	whoami(1)
whodo - which users are doing what	whodo(1M)
whois - Internet user name directory service	whois(1)
wide character back into input stream, push	ungetwc(3C)
wide character from a stream file, get	getwc(3C)

Index

All Volumes

Description

Entry Name(Section)

wide character string and rendition, get from a <code>cchar_t</code>	<code>getcchar</code> (3X)
wide character string and rendition, set <code>cchar_t</code>	<code>setcchar</code> (3X)
wide character string operations	<code>wcstring</code> (3C)
wide character string to double-precision number, convert	<code>wctod</code> (3C)
wide character string to long integer, convert	<code>wctol</code> (3C)
wide character, generate printable representation of	<code>wunctrl</code> (3X)
wide character, get from a terminal	<code>get_wch</code> (3X)
wide character, put on a stream	<code>putwc</code> (3C)
wide characters, an array of, and function key codes, get from a terminal	<code>getn_wstr</code> (3X)
wide characters, string of, input from a window	<code>innwstr</code> (3X)
wide characters, translate to uppercase or lowercase	<code>wconvc</code> (3C)
wide string from a stream file, input	<code>fgetws</code> (3C)
wide strings, concatenate two	<code>wcstring</code> (3C)
wide-character string, convert date and time to	<code>wcsftime</code> (3C)
wide-character string, insert into a window	<code>ins_nwstr</code> (3X)
width (in bits) of data path, set	<code>io_width_ctl</code> (3I)
<code>winch()</code> - input a single-byte character and rendition from a window	<code>inch</code> (3X)
<code>winchnstr()</code> - input an array of single-byte characters and renditions from a window	<code>inchnstr</code> (3X)
<code>winchstr()</code> - input an array of single-byte characters and renditions from a window	<code>inchnstr</code> (3X)
window and cursor coordinates, get additional	<code>getbegyx</code> (3X)
window attribute control functions	<code>attr_get</code> (3X)
window attribute control functions, restricted	<code>attroff</code> (3X)
window attributes, set and clear	<code>standend</code> (3X)
window coordinate transformation, define	<code>mvderwin</code> (3X)
window creation function	<code>derwin</code> (3X)
window creation functions	<code>newwin</code> (3X)
window cursor location functions	<code>move</code> (3X)
window refresh control function	<code>touchwin</code> (3X)
window refresh control functions	<code>is_linetouched</code> (3X)
window refreshed after echo single-byte character and rendition	<code>echochar</code> (3X)
window, change renditions of characters in a window	<code>chgat</code> (3X)
window, clear	<code>clear</code> (3X)
window, clear from cursor to end of window	<code>clrtoebot</code> (3X)
window, convert formatted input from	<code>mvscanw</code> (3X)
window, convert formatted input	<code>vwscanw</code> (3X)
window, convert formatted input	<code>vw_scanw</code> (3X)
window, copy a region of window	<code>copywin</code> (3X)
window, current	<code>curscr</code> (3X)
window, default	<code>stdscr</code> (3X)
window, delete	<code>delwin</code> (3X)
window, delete or insert lines into	<code>insdelln</code> (3X)
window, dump to and reload from a file	<code>getwin</code> (3X)
window, duplicate	<code>dupwin</code> (3X)
window, get cursor and window coordinates	<code>getyx</code> (3X)
window, input a complex character and rendition from	<code>in_wch</code> (3X)
window, input a multi-byte character string from	<code>innstr</code> (3X)
window, input a single-byte character and rendition from	<code>inch</code> (3X)
window, input a string of wide characters from	<code>innwstr</code> (3X)
window, input an array of complex characters and renditions from	<code>in_wchnstr</code> (3X)
window, insert a complex character and rendition into	<code>ins_wch</code> (3X)
window, insert a multi-byte character into	<code>insnstr</code> (3X)
window, insert a single-byte character and rendition into	<code>insch</code> (3X)
window, insert a wide-character string into	<code>ins_nwstr</code> (3X)
window, insert lines into	<code>insertln</code> (3X)
window, move	<code>mvwin</code> (3X)
window, print formatted output in	<code>mvprintw</code> (3X)
window, print formatted output	<code>vwprintw</code> (3X)
window, print formatted output	<code>vw_printw</code> (3X)
window, refresh immediately after writing a complex character	<code>echo_wchar</code> (3X)
window, scroll a curses window	<code>scroll</code> (3X)
window, scroll, enhanced curses	<code>sctrl</code> (3X)

Description	Entry Name(Section)
window, synchronise with its parents or children	syncok(3X)
windows, copy overlapped windows	overlay(3X)
winnstr() - input a multi-byte character string from a window	innstr(3X)
winnwstr() - input a string of wide characters from a window	innwstr(3X)
winsch() - insert a single-byte character and rendition into a window	insch(3X)
winsdelln() - delete or insert lines into a window	insdelln(3X)
winsertln() - insert lines into a window	insertln(3X)
winsnstr() - insert a multi-byte character into a window	insnstr(3X)
winsstr() - insert a multi-byte character into a window	insnstr(3X)
winstr() - input a multi-byte character string from a window	innstr(3X)
wins_nwstr() - insert a wide-character string into a window	ins_nwstr(3X)
wins_wch() - insert a complex character and rendition into a window	ins_wch(3X)
wins_wstr() - insert a wide-character string into a window	ins_nwstr(3X)
winwstr() - input a string of wide characters from a window	innwstr(3X)
win_wch() - input a complex character and rendition from a window	in_wch(3X)
win_wchnstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
win_wchstr() - input an array of complex characters and renditions from a window	in_wchnstr(3X)
wmove() - window cursor location functions	move(3X)
wnoutrefresh() - refresh windows and lines	doupdate(3X)
word expansions, perform	wordexp(3C)
word from a stream file, get character or data	getc(3S)
word or character, put on a stream	putc(3S)
wordexp() - perform word expansions	wordexp(3C)
wordfree() - free memory associated with word expansions	wordexp(3C)
words in a file, count	wc(1)
words, find hyphenated	hyphen(1)
working directory	glossary(9)
working directory name, print	pwd(1)
working directory, change	cd(1)
working directory, change	chdir(2)
working directory, get path-name of current	getcwd(3C)
working directory, get pathname of current	getwd(3C)
wprintw() - print formatted output in window	mvpprintw(3X)
wredrawln() - line update status functions	redrawwin(3X)
wrefresh() - refresh windows and lines	doupdate(3X)
write(1) messages from other users to terminal, deny or permit	mesg(1)
write a character rendition and immediately refresh the pad	pechochar(3X)
write a complex character and immediately refresh the window	echo_wchar(3X)
write a message simultaneously to all users	wall(1M)
write a null-terminated string on a stream	puts(3S)
write a null-terminated wide string on a stream	putws(3C)
write audit record for self-auditing process	auditwrite(2)
write end-of-file marks on magnetic tape	mt(1)
write - interactively write (talk) to another user	write(1)
write LIF volume header on file	lifinit(1)
write password file entry	putpwent(3C)
write STREAMS event trace messages to standard output	strace(1M)
write to all users over a network	rwall(1M)
write to specified remote machines	rwall(3N)
write() - write contiguous data to a file	write(2)
write, asynchronous start	aio_write(2)
write/read file pointer, move	lseek(2)
writev() - write noncontiguous data to a file	write(2)
writing, open file for	open(2)
wscanw() - convert formatted input from a window	mvscanw(3X)
wscrl() - scroll the window, enhanced curses	scrl(3X)
wsetscrreg() - terminal output control functions	clearok(3X)
wstandend() - set and clear window attributes	standend(3X)
wstandout() - set and clear window attributes	standend(3X)
wsyncdown() - synchronise a window with its parents or children	syncok(3X)
wsyncup() - synchronise a window with its parents or children	syncok(3X)

Index

All Volumes

Description	Entry Name(Section)
wtimeout() - control blocking on input	notimeout(3X)
wtmp file	login(1)
wtmp(), utmp(), btmp() - utmp, wtmp, btmp user accounting file entry format	utmp(4)
wtmp() or utmp() file, access	getut(3C)
wtmpfix - manipulate connect accounting records	fwtmp(1M)
wtouchln() - window refresh control functions	is_linetouched(3X)
wunctrl() - generate printable representation of a wide character	wunctrl(3X)
wvline() - draw lines from single-byte characters and renditions	hline(3X)
wvline_set() - draw lines from complex characters and renditions	hline_set(3X)
Wyse60 terminal	cue(1)
Wyse60 terminal	sam(1M)
Wyse60 terminal	swinstall(1M)
Wyse60 terminal	swremove(1M)
X.25 line, get	getx25(1M)
X.25 switched virtual circuit, clear	clrsvc(1M)
X/Open Conformance Statement Questionnaire, pointer to manpage, x_open_800	x_open(5)
X/Open Networking	xopen_networking(7)
X/OPEN Transport Interface - XTI, accept a connect request issued by a transport user	t_accept(3)
X/OPEN Transport Interface - XTI, acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
X/OPEN Transport Interface - XTI, allocate library structure	t_alloc(3)
X/OPEN Transport Interface - XTI, bind address to transport endpoint	t_bind(3)
X/OPEN Transport Interface - XTI, close transport endpoint	t_close(3)
X/OPEN Transport Interface - XTI, disable transport endpoint	t_unbind(3)
X/OPEN Transport Interface - XTI, error message function	t_error(3)
X/OPEN Transport Interface - XTI, establish connection with another transport user	t_connect(3)
X/OPEN Transport Interface - XTI, establish transport endpoint	t_open(3)
X/OPEN Transport Interface - XTI, free library structure	t_free(3)
X/OPEN Transport Interface - XTI, get current state	t_getstate(3)
X/OPEN Transport Interface - XTI, get protocol address	t_getprotaddr(3)
X/OPEN Transport Interface - XTI, get protocol-specific service information	t_getinfo(3)
X/OPEN Transport Interface - XTI, initiate orderly release at transport endpoint	t_sndrel(3)
X/OPEN Transport Interface - XTI, listen for connect request	t_listen(3)
X/OPEN Transport Interface - XTI, look at current event on transport endpoint	t_look(3)
X/OPEN Transport Interface - XTI, manage options for transport endpoint	t_optmgmt(3)
X/OPEN Transport Interface - XTI, produce error message string	t_strerror(3)
X/OPEN Transport Interface - XTI, receive confirmation from connect request	t_rcvconnect(3)
X/OPEN Transport Interface - XTI, receive data over connection	t_rcv(3)
X/OPEN Transport Interface - XTI, receive data unit from remote transport provider user	t_rcvudata(3)
X/OPEN Transport Interface - XTI, receive error information from unit data error indication	t_rcvuderr(3)
X/OPEN Transport Interface - XTI, retrieve disconnect information	t_rcvdis(3)
X/OPEN Transport Interface - XTI, send data or expedited data over a connection	t_snd(3)
X/OPEN Transport Interface - XTI, send data unit to transport user	t_sndudata(3)
X/OPEN Transport Interface - XTI, send user-initiated disconnect request	t_snddis(3)
X/OPEN Transport Interface - XTI, synchronize transport library for transport endpoint	t_sync(3)
X/OPEN Transport Layer Interface - TLI, accept a connect request issued by a transport user	t_accept(3)
X/OPEN Transport Layer Interface - TLI, acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
X/OPEN Transport Layer Interface - TLI, allocate library structure	t_alloc(3)
X/OPEN Transport Layer Interface - TLI, bind address to transport endpoint	t_bind(3)
X/OPEN Transport Layer Interface - TLI, close transport endpoint	t_close(3)
X/OPEN Transport Layer Interface - TLI, disable transport endpoint	t_unbind(3)
X/OPEN Transport Layer Interface - TLI, error message function	t_error(3)
X/OPEN Transport Layer Interface - TLI, establish connection with another transport user	t_connect(3)
X/OPEN Transport Layer Interface - TLI, establish transport endpoint	t_open(3)
X/OPEN Transport Layer Interface - TLI, free library structure	t_free(3)
X/OPEN Transport Layer Interface - TLI, get current state	t_getstate(3)
X/OPEN Transport Layer Interface - TLI, get protocol-specific service information	t_getinfo(3)
X/OPEN Transport Layer Interface - TLI, initiate orderly release at transport endpoint	t_sndrel(3)
X/OPEN Transport Layer Interface - TLI, listen for connect request	t_listen(3)
X/OPEN Transport Layer Interface - TLI, look at current event on transport endpoint	t_look(3)

Description	Entry Name(Section)
X/OPEN Transport Layer Interface - TLI, manage options for transport endpoint	t_optmgmt(3)
X/OPEN Transport Layer Interface - TLI, receive confirmation from connect request	t_rcvconnect(3)
X/OPEN Transport Layer Interface - TLI, receive data over connection	t_rcv(3)
X/OPEN Transport Layer Interface - TLI, receive data unit from remote transport provider user	t_rcvdata(3)
X/OPEN Transport Layer Interface - TLI, receive error information from unit data error indication	t_rcvuderr(3)
X/OPEN Transport Layer Interface - TLI, retrieve disconnect information	t_rcvdis(3)
X/OPEN Transport Layer Interface - TLI, send data or expedited data over a connection	t_snd(3)
X/OPEN Transport Layer Interface - TLI, send data unit to transport user	t_sndudata(3)
X/OPEN Transport Layer Interface - TLI, send user-initiated disconnect request	t_snddis(3)
X/OPEN Transport Layer Interface - TLI, synchronize transport library for transport endpoint	t_sync(3)
xargs - construct argument list(s) and execute command	xargs(1)
xd - hexadecimal file dump	od(1)
xdr - library routines for external data representation	xdr(3N)
XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr, library routines for external data representation	xdr(3N)
xdr, library routines for external data representation	xdr_admin(3N)
xdr, library routines for external data representation	xdr_complex(3N)
xdr, library routines for external data representation	xdr_simple(3N)
xdr, library routines for external data representation stream creation	xdr_create(3N)
xdrmem_create() - library routines for external data representation stream creation	xdr_create(3N)
xdrrec_create() - library routines for external data representation stream creation	xdr_create(3N)
xdrrec_endofrecord() - library routines for external data representation	xdr_admin(3N)
xdrrec_eof() - library routines for external data representation	xdr_admin(3N)
xdrrec_readbytes() - library routines for external data representation	xdr_admin(3N)
xdrrec_skiprecord() - library routines for external data representation	xdr_admin(3N)
xdrstdio_create() - library routines for external data representation stream creation	xdr_create(3N)
xdr_accepted_reply() - write noncontiguous data to a file	write(2)
xdr_accepted_reply() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_admin() - library routines for external data representation	xdr_admin(3N)
xdr_array() - library routine for external data representation	xdr_complex(3N)
xdr_authsys_parms() - write noncontiguous data to a file	write(2)
xdr_authsys_parms() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_authunix_parms() - write noncontiguous data to a file	rpc_soc(3N)
xdr_bool() - library routines for external data representation	xdr_simple(3N)
xdr_bytes() - library routine for external data representation	xdr_complex(3N)
xdr_callhdr() - write noncontiguous data to a file	write(2)
xdr_callhdr() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_callmsg() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_callmsg() - write noncontiguous data to a file	write(2)
xdr_char() - library routines for external data representation	xdr_simple(3N)
xdr_complex() - library routine for external data representation	xdr_complex(3N)
xdr_control() - library routines for external data representation	xdr_admin(3N)
xdr_create() - library routines for external data representation stream creation	xdr_create(3N)
xdr_destroy() - library routines for external data representation stream creation	xdr_create(3N)
xdr_double() - library routines for external data representation	xdr_simple(3N)
xdr_enum() - library routines for external data representation	xdr_simple(3N)
xdr_float() - library routines for external data representation	xdr_simple(3N)
xdr_free() - library routines for external data representation	xdr_simple(3N)
xdr_getpos() - library routines for external data representation	xdr_admin(3N)
xdr_hyper() - library routines for external data representation	xdr_simple(3N)
xdr_inline() - library routines for external data representation	xdr_admin(3N)
xdr_int() - library routines for external data representation	xdr_simple(3N)
xdr_long() - library routines for external data representation	xdr_simple(3N)
xdr_longlong_t() - library routines for external data representation	xdr_simple(3N)
xdr_opaque() - library routine for external data representation	xdr_complex(3N)
xdr_opaque_auth() - write noncontiguous data to a file	write(2)
xdr_opaque_auth() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_pointer() - library routine for external data representation	xdr_complex(3N)
xdr_quadruple() - library routines for external data representation	xdr_simple(3N)
xdr_reference() - library routine for external data representation	xdr_complex(3N)

Index

All Volumes

Description	Entry Name(Section)
xdr_rejected_reply() - write noncontiguous data to a file	write(2)
xdr_rejected_reply() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_replymsg() - write noncontiguous data to a file	write(2)
xdr_replymsg() - XDR library routines for remote procedure calls	rpc_xdr(3N)
xdr_setpos() - library routines for external data representation	xdr_admin(3N)
xdr_short() - library routines for external data representation	xdr_simple(3N)
xdr_simple() - library routines for external data representation	xdr_simple(3N)
xdr_sizeof() - library routines for external data representation	xdr_admin(3N)
xdr_string() - library routine for external data representation	xdr_complex(3N)
xdr_union() - library routine for external data representation	xdr_complex(3N)
xdr_u_char() - library routines for external data representation	xdr_simple(3N)
xdr_u_hyper() - library routines for external data representation	xdr_simple(3N)
xdr_u_int() - library routines for external data representation	xdr_simple(3N)
xdr_u_long() - library routines for external data representation	xdr_simple(3N)
xdr_u_longlong_t() - library routines for external data representation	xdr_simple(3N)
xdr_u_short() - library routines for external data representation	xdr_simple(3N)
xdr_vector() - library routine for external data representation	xdr_complex(3N)
xdr_void() - library routines for external data representation	xdr_simple(3N)
xdr_wrapstring() - library routine for external data representation	xdr_complex(3N)
XMODEM-protocol file transfer program	umodem(1)
xntpd - Network Time Protocol daemon	xntpd(1M)
xopen_networking - X/Open Networking	xopen_networking(7)
xprt_register() - library routines for registering servers	rpc_svc_reg(3N)
xstr - extract strings from C programs to implement shared strings	xstr(1)
xtab - directories to export to NFS clients	exports(4)
XTI function, accept a connect request issued by a transport user	t_accept(3)
XTI function, acknowledge receipt of orderly release indication at transport endpoint	t_rcvrel(3)
XTI function, allocate a library structure	t_alloc(3)
XTI function, bind address to transport endpoint	t_bind(3)
XTI function, close transport endpoint	t_close(3)
XTI function, disable transport endpoint	t_unbind(3)
XTI function, error message function	t_error(3)
XTI function, establish connection with another transport user	t_connect(3)
XTI function, establish transport endpoint	t_open(3)
XTI function, free library structure	t_free(3)
XTI function, get current state	t_getstate(3)
XTI function, get protocol address	t_getprotaddr(3)
XTI function, get protocol-specific service information	t_getinfo(3)
XTI function, initiate orderly release at transport endpoint	t_sndrel(3)
XTI function, listen for connect request	t_listen(3)
XTI function, look at current event on transport endpoint	t_look(3)
XTI function, manage options for transport endpoint	t_optmgmt(3)
XTI function, produce error message string	t_strerror(3)
XTI function, receive confirmation from connect request	t_rcvconnect(3)
XTI function, receive data over connection	t_rcv(3)
XTI function, receive data unit from remote transport provider user	t_rcvudata(3)
XTI function, receive error information from unit data error indication	t_rcvuderr(3)
XTI function, retrieve disconnect information	t_rcvdis(3)
XTI function, send data or expedited data over a connection	t_snd(3)
XTI function, send data unit to transport user	t_sndudata(3)
XTI function, send user-initiated disconnect request	t_snddis(3)
XTI function, synchronize transport library for transport endpoint	t_sync(3)
x_open(5) - pointer manual entry for X/Open Conformance Statement Questionnaire	x_open(5)
x_open_800, X/Open Conformance Statement Questionnaire	x_open(5)
y0(), y1(), yn() - Bessel functions of the second kind	y0(3M)
y1() - Bessel function	y0(3M)
yes - repetitively affirmative responses	yes(1)
yield frequency attribute, get or set	pthread_mutexattr_getspin_np(3T)
yn() - Bessel function	y0(3M)
ypbind - Network Information Service (NIS) binder processes	ypserv(1M)
ypcat - print values in Network Information Service map	ypcat(1)

Description	Entry Name(Section)
<code>ypclnt()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yperr_string()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>ypfiles</code> - Network Information Service database and directory structure	<code>ypfiles(4)</code>
<code>ypinit</code> - build and install Network Information Service databases	<code>ypinit(1M)</code>
<code>ypmake</code> - create or rebuild Network Information Service database	<code>ypmake(1M)</code>
<code>ypmatch</code> - print the values of selected keys in Network Information Service map	<code>ypmatch(1)</code>
<code>yppasswd</code> - change login password in Network Information System (NIS)	<code>yppasswd(1)</code>
<code>yppasswd()</code> - update user password in Network Information Service	<code>yppasswd(3N)</code>
<code>yppasswdd</code> - daemon for modifying Network Information Service <code>passwd</code> database	<code>yppasswdd(1M)</code>
<code>yppoll</code> - query an NIS server for information about an NIS map	<code>yppoll(1M)</code>
<code>ypprot_err()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yppush</code> - force propagation of a Network Information Service database	<code>yppush(1M)</code>
<code>ypserv</code> - Network Information Service (NIS) server processes	<code>ypserv(1M)</code>
<code>ypset</code> - bind to a particular Network Information Service server	<code>ypset(1M)</code>
<code>ypupdate()</code> - changes NIS information	<code>ypupdate(3C)</code>
<code>ypupdated, rpc.yupdated</code> - server for changing NIS information	<code>ypupdated(1M)</code>
<code>ypwhich</code> - list which host is Network Information System server or map master	<code>ypwhich(1)</code>
<code>ypxfr</code> - transfer NIS database from NIS server to local node	<code>ypxfr(1M)</code>
<code>ypxfrd</code> - Network Information Service (NIS) transfer processes	<code>ypserv(1M)</code>
<code>ypxfr_1perday</code> - transfer NIS database from NIS server to local node	<code>ypxfr(1M)</code>
<code>ypxfr_1perhour</code> - transfer NIS database from NIS server to local node	<code>ypxfr(1M)</code>
<code>ypxfr_2perday</code> - transfer NIS database from NIS server to local node	<code>ypxfr(1M)</code>
<code>yp_all()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_bind()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_first()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_get_default_domain()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_master()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_match()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_next()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_order()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>yp_unbind()</code> - Network Information Service client interface	<code>ypclnt(3C)</code>
<code>zcat, compress, uncompress</code> - compress or expand data	<code>compress(1)</code>
zero-length file, create	<code>cat(1)</code>
zero-length file, create	<code>cp(1)</code>
zero-length file, create	<code>null(7)</code>
zero-length file, create	<code>touch(1)</code>
zombie process	<code>glossary(9)</code>
<code>_authdes_getucrd()</code> - library routines for secure remote procedure calls	<code>secure_rpc(3N)</code>
<code>_ldcvct(), _ldfcvt(), _ldgcvct()</code> - convert long double to string	<code>ldcvct(3C)</code>
<code>_longjmp()</code> - restore stack environment after non-local goto	<code>setjmp(3C)</code>
<code>_nis_map_group()</code> - NIS+ group manipulation functions	<code>nis_groups(3N)</code>
<code>_pututline()</code> - update or create entry in a <code>utmp()</code> file	<code>getut(3C)</code>
<code>_setjmp()</code> - save stack environment for non-local goto	<code>setjmp(3C)</code>
"floppy" or flexible disk device driver	<code>floppy(7)</code>